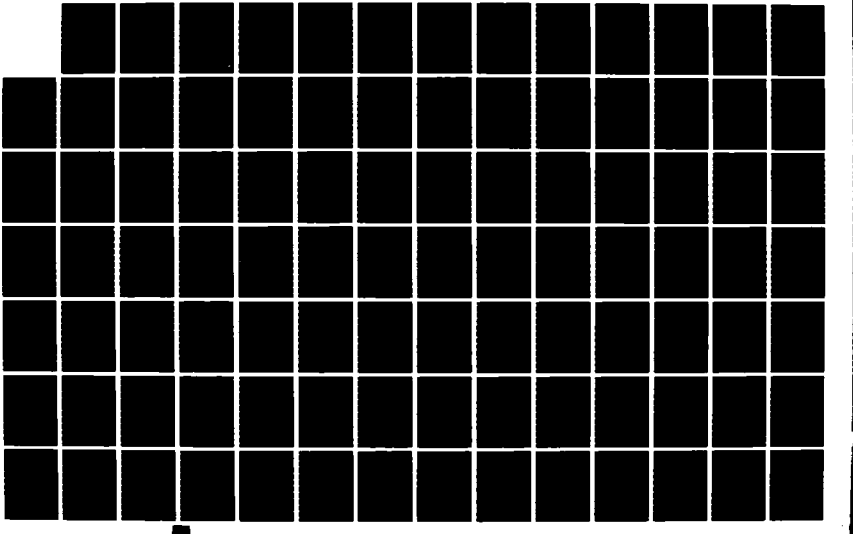
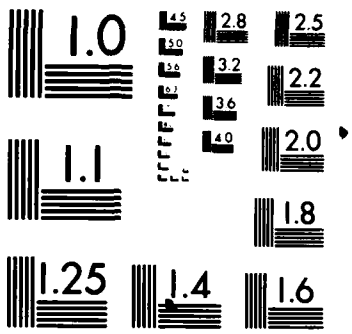
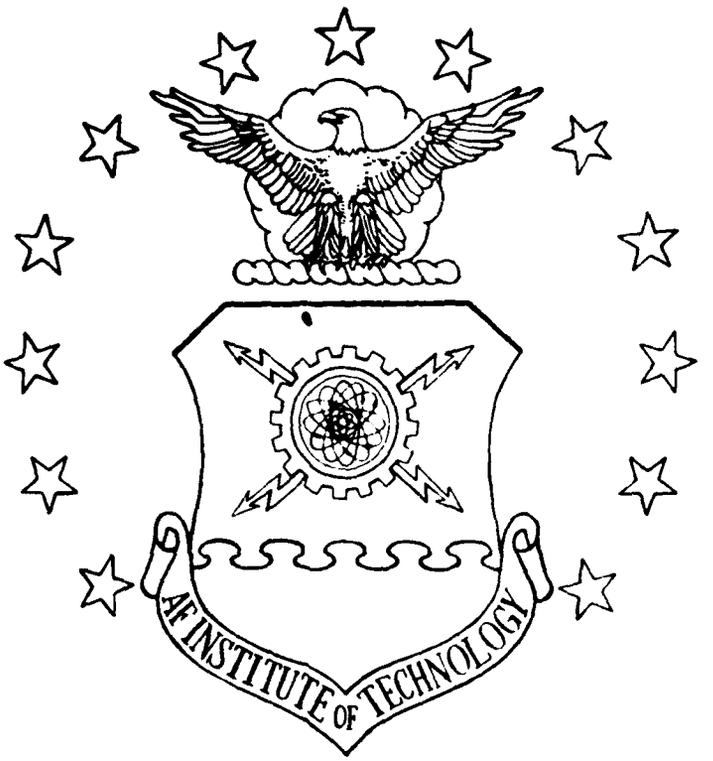
AD-A172 454 DESIGN AND IMPLEMENTATION OF DATA BASE MANAGEMENT SYSTEM FOR THE ORGANIZA..(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGI.. G A ELSHARAWY 1/2

UNCLASSIFIED DEC 86 AFIT/GCS/ENG/86J-4 F/G 5/9 NL







DESIGN AND IMPLEMENTATION OF DATA BASE MANAGEMENT SYSTEM FOR THE ORGANIZATION STRUCTURE OF THE EGYPTIAN ARMED FORCES

THESIS

Gaber A. Elsharawy
Lt. Col. Egypt Army

AFIT/GCS/ENG/86J-4

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/GCS/ENG/86J-4

DESIGN AND IMPLEMENTATION OF DATA BASE MANAGEMENT SYSTEM FOR THE ORGANIZATION STRUCTURE OF THE EGYPTIAN ARMED FORCES

THESIS

Gaber A. Elsharawy
Lt. Col. Egypt Army

AFIT/GCS/ENG/86J-4

Approved for public release; distribution unlimited.

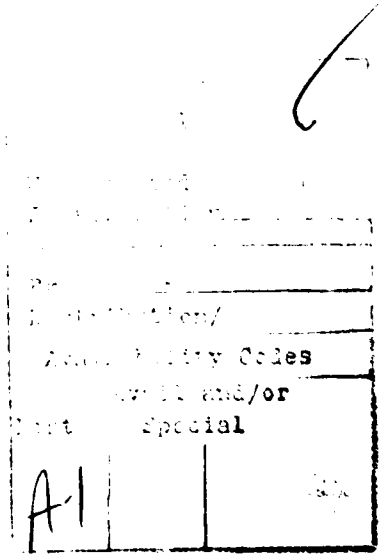
AFIT/GCS/ENG/86J-4

DESIGN AND IMPLEMENTATION OF DATA BASE MANAGEMENT SYSTEM FOR THE ORGANIZATION STRUCTURE OF THE EGYPTIAN ARMED FORCES

THESIS

Presented to the Faculty of the School of Engineering
of the Air Force Institute of Technology
Air University
In Partial Fulfillment of the
Requirements for the Degree of
Master of Science in Computer Systems

Gabe: Ahmed Elsharawy, Dip. Eng.
Lt. Colonel, Egyptian Army



December 1986

Preface

The purpose of this study is two fold. The first purpose is to design a data base management system model for the organization structure of the Egyptian Armed Force as a part of the integrated information system. The second purpose is to implement that model as an applicable system ready to serve in the Egyptian Armed Force.

I would like to acknowledge the support and encouragement that I received from my thesis advisor Dr. Henry Potoczny and from the committee members, Dr. Thomas Hartrum, Lt. Col. Richard Gross and Maj. John Stibravy.

Finally, I would like to thank my wife, Nagwa, and the rest of my family, for their constant understanding, support, and encouragement without which I could not have finished this work.

Gaber A. Elsharawy

Table of Contents

List of Figures

AFIT/GCS/ENG/86J-4

Abstract

The organization structure system is a part of the integrated information system of the Eygptian Armed Forces. It manipulates the units organization data (department, unit level, type, balance, armament, personnel, plan(s), percentage of completeness, location and position in the armed force tree). An initial survey for the problem is done. The design of the system is done through the E-R model and the functional dependency is defined. We choose the relational data base model for its advantages like data independent and simple data manipulation, over the other two DBMS models (hierarchical and network models). The system is designed in ten relations and the implementation is done through ingress using C programming language with equal (embedded ingres in C). We present several examples of queries that the system can support. An algorithm for collecting the units commanded by a particular unit is presented. The implementation includes the data definition of the ten relations in ingres. The data base editing program is presented which consists of 23 modules. This program is able to perform the addition, modification, deletion, and retrieval of units data keeping the data base in consistent state. The problems of recovery, concurrency, security and data integrity are also discussed.

DESIGN AND IMPLEMENTATION OF DATA BASE MANAGEMENT SYSTEM FOR THE ORGANIZATION STRUCTURE OF THE EGYPTIAN ARMED FORCES

I. Introduction

Background

In Egypt it's highly recommended that a national integrated information system be built. The Information System Department did a lot of work in the analysis and the design phase of the integrated information system. Building the integrated information system for the Egyptian Armed Force is very essential. This integrated system will support the dynamic decision making process in all military areas. There is a lot of work done in the area of investigation and analysis of the integrated information system. There are many applications implemented as file system projects, but there's no application designed in data base management system yet.

Problem

The purpose of this thesis effort is to design and implement a data base management system for the organization structure of the Egyptian Armed Forces as a subsystem of the integrated information system keeping in consideration data base integrity and system interfaces to other systems/ subsystems.

Scope

The scope of this effort was limited to the design and implementation of a prototype data base management system for the organization structure of the Egyptian Armed Forces. This newly designed system is able to answer any query related to the organization structure data. An Editing Program for the unit data is implemented.

Approach

The approach for this project follows:

1. General study of the Egyptian Armed Forces structure.
2. Description of the integrated information system.
3. Design of entity relationship model for the new designed system.
4. Choice of data base management system model.
5. Design of logical data base.
6. Physical data base representation.
7. Design and implementation of the editing program.
8. Discussion about the system related problems (recovery, concurrency, security, integrity, and system interfaces).

II. An Overview of the Eyptian Armed Force

Introduction

The Egyptian Armed Force formation consists of six main entities:

1. Arms and Equipments
2 Manpower
3. Unit Formation
4. Supply
5. Facilities
6. Transportation

Actually no single entity can stand by itself.

To get the system to work, we must assign the suitable arms and equipment with a well trained manpower to a suitable, well organized unit, and secure supplies, facilities, and transportation. That lets a single unit work in reasonable efficiency. But our goal is to let the whole armed force work in harmony with high efficiency and effectiveness in both war and peace time.

No doubt, the only way to reach the required degree of efficiency of the armed force that is equipped with a modern arms and equipment is through an integrated information system.

Developing a computerized national information system in Egypt is under investigation. The Egyptian Armed Force is

the leader in that field. The Ministry of Defense (MOD) is always eager to help other ministries build their own computerized information system.

Before attempting to design the computer system we will make a brief description of the Egyptian Armed Force from the point of view of the six main entities mentioned above.

Description of Egyptian Armed Forces

The Egyptian Armed Forces consists of five main branches:

1. Army
2. Navy
3. Air Force
4. Air Defense
5. Border Guard

Actually each branch may operate independently and has its own command. In the Army, each field army, or military area, has its own command and its own budget, but it is not completely independent. For example, logistic support is centralized for the whole armed force; also, personnel supplies with draftees is centralized (it is permitted to let the enlisted person join the branch of his choice). The description of the Egyptian Armed Force will focus on the six main entities:

1. Arms and Equipment
2. Manpower
3. Unit Formation
4. Supply
5. Facilities
6. Transportation

Arms and Equipments (A & E)

The Egyptian Armed Force has a wide variety of arms and equipment that maybe classified into four main categories:

1. U. S. A & E
2. Eastern A & E
3. European A & E
4. Egyptian A & E

The different sources of A & E create difficulties in classification and codification processes for the arms and equipment (complete set and parts). Several studies were done on the existing inventory systems, supply systems, and technical support system.

The studies yield a decision taken by the Egyptian Ministry of Defense (MOD) to use the NATO system of classification and codification for complete arms and equipment and parts as the primary base of the new Egyptian computerized inventory system (as a subsystem of the logistic support system). To show the variety of A & E, the following are the types of arms used by the Egyptian armor and artillery as estimated in the Air Force Magazine,

December 1984.

Main Fighting Tanks

T-54, T-55, T-62, AM-60(M-60 A3), PT-76

Armored Fighting Vehicles

BRDM-1, BRDM-2, Scout ARS, BMP-1, PMP-600P, Compat Vehicles, OT-62, Walid,FAhd, BTR-40, BTR-50, BTR-60, BTR-152 armor carriers, M 113 A2 armored carrier.

Artillery

Guns: 85 mm, M-1955, SU100-100 mm, D-30, 122 mm, M-46-130 mm, SU-152, S-23, 180 mm.

Howitzers: M-1938 122 mm, M-1943 192 mm.

Mortars: 120 mm, 160 mm, 240 mm.

Multi-Rocket launchers

122 mm, 132 mm, 140 mm, 240 mm.

Surface to Surface Missiles

Frog-7, Scud-B

Anti-Tank

Recoiless Launchers: 57 mm, 76 mm, 100 mm, 82 mm, B11-107 mm

Guided Weapons

Sagger, Snapper, Swatter, Milan, Beeswing, Swingfire, Tow

Air Defense

Self Propelled Aircraft Guns: ZS U-23-4, ZS U-57-Z

Surface to Air Missile: SA-6, SA-7, SA-9, Carstal, Skyguard (6:122)

(This is not included in Air Defense Command)

From these arms, we can find the eastern arms like the T-62 tank, the U.S. arms like the M-60 tank, the European arms like Crotal Air Defense System, and the Egyptian arms like Walid armored vehicle.

For all the A & E, we need a classification that can meet the requirements of the new designed system. This will be discussed later.

Manpower

The manpower may be divided into three main categories:

a. Professionals:

That includes active duty officers, non commissioned officers, and civilians who are working for the Armed Forces.

b. Draftees:

The draftees are men recruited to serve in the armed forces a mandatory service for a certain period of time. Four years for education below high school, three years for high school graduates and one year for education higher than high school. That is three years' average. The draftees recruited as soldiers, except for the draftees who have education higher than high school, may be recruited as officers. In this case, they must stay two years in the service.

c. Reserve:

All draftees must stay in the reserve service a certain period of time. The reserve man may be reassigned to the same unit he was serving in, or to another unit. The reserve is mobilized periodically for training.

Another classification of manpower which is mainly used for planning and for unit establishment tables is:

a. Officer:

This includes all types of officers whatever their ranks, specializations, type of service.

b. Secretary:

This includes all military personnel who are working in jobs like clerk, administration, finance, typing, document handling, librarians, bookkeeping, etc.

c. Commissions:

This includes all military personnel whose jobs are to operate in an arm or equipment except for driving and technical maintenance, and also includes military drill personnel.

d. Driver:

This includes all types of drivers: truck drivers, tank drivers, clerk drivers, bulldozer drivers, etc.

e. Technician:

This includes all military personnel whose work is in technical maintenance, technical inspection, and also personnel who work in technical training jobs.

f. Craftees:

This includes all craft jobs like tailors, carpenters, painters, operators, etc.

Unit Formation.

There are different types of units in the Egyptian Armed Forces. The classification of the unit types is done according to the main mission and level of command. The unit name and class does not necessarily reflect its level of command. For example, one company may be commanded directly by a field army command, another is related to battalion then brigade, then division, then to a field army command. The following is a brief description of the unit main types:

Ministry of Defense (MOD).

This is the highest command level in the Egyptian Armed Forces and it commands all other units.

Main Branches.

There are five main branches (army, navy, air force, air defense, and border guards). Each branch has a single command. Except for the army, there is a single command for each field army or military area. The main branch is commanded directly by the MOD.

Field Armies and Military Areas.

Each field army or military area is a complete formation of army units. (There may be some Air Defense

units included in the formation established for a certain mission(s)). A field army or military area is commanded directly by the MOD.

Authorities.

Each authority is completely responsible for a main mission(s) in the armed forces. For example, the training authority is responsible for planning and follow-up of all types of training in the armed forces starting from primary military drill to the specialized training and graduate programs needed. The logistic authority is completely responsible for all logistic affairs for all armed forces. The authorities are commanded directly by MOD.

Centralized Departments.

Each centalized department represents a branch of service for the military personnel and commanded directly by MOD. Infantry departments, armor departments, signal corp departments, are examples of the centralized departments. Each specialized department is responsible for supplying the armed force units and formations with personnel. For example, the armor department is responsible for supplying the armed force units and formation with armor personnel according to each unit establishment table and percentage of completeness (this will be discussed later).

Each centralized department is responsible for supporting, reviewing, and inspecting of the unit of the

same branch of service. For example, the infantry department is responsible for completeness of all personnel in all infantry units. That means it should know the requirements of each unit from each branch of service to reach the required efficiency. In other words, the infantry department should know types and quantities of personnel needed from the infantry department and from other departments to complete an infantry unit according to its establishment table and percentage of completeness. Some specialized departments are completely responsible to supply arms and equipment related to the same branch of service, some are not.

For example, the armor department is responsible for supply, technical support, training personnel, and inspection of all armor equipment in armor units and also other units. On the other hand, the infantry department is not responsible for technical support of infantry weapons. A good example of showing how the centralized departments cooperate to support one formation is a mechanized infantry brigade.

The infantry department is the sponsor of the mechanized infantry brigade and should help it get its needs from other departments. For example, the infantry department provides it with personnel only. The armor department provides it with armo. carriers, technical

equipment, and parts needed for its operation. The armor department is also providing it with armor drivers and with technical support personnel. The brigade also contains a tank battalion (armor unit). The fuel department provides the brigade with the needed fuel and oil and also personnel needed for handling and storing that fuel. The engineering department provides the brigade with engineering equipment and personnel. The air defense command provides the brigade with complete air defence battalion (only personnel). The arms and ammunition department provides the brigade with all needed types of arms and ammunition and related equipment, and also provides it with personnel needed for handling and technical maintenance of arms and ammunitions. Also, other departments like the signal corp, chemical, artillery, electronic warfare, food, clothing, etc. provide the brigade with different needs. From the above example, we can recognize that there are many departments that are responsible for providing each army unit with its needs. To keep the units and the formations free from contacting all departments, there is a sponsor department for each unit and formation that should do what is necessary to let the unit or the formation reach the required degree of completeness.

Non Centralized Departments.

The non centralized departments are not commanded directly by the MOD and/or do not represent a branch of service. Some examples of these follow.

The clothing department represents a branch of service but is commanded by the logistic authority. The information systems department is commanded by MOD but does not represent a branch of service. The recruiting department is commanded directly by organization and management authority and does not represent a branch of service. A non centralized department maybe a sponsor of a certain type of unit.

Formation Units.

In the army, formation units start from the field army, division, brigade, battalion, company, platoon, and team. In the air force it starts from air base, air brigade, air port, and squadron.

Other types of units such as workshop, depot, store, weather station, etc., should be considered also.

Supply.

There are three main types of supplies:

- Arms and equipment supplies.
- Personnel supplies.
- Consumer material supplies.

Arms and Equipment Supplies.

Arms and equipment supply is done in both peace and war time for several reasons:

- New unit establishment.
- Replacement of damaged or destroyed equipment.
- Replacement plan by another type of arm/ equipment.
- Completion of units.

Personnel Supplies.

As the units do not have to be completed 100% during peace time, personnel supply with the needed specialization of personnel will function during peace time for different reasons:

- New established unit.
- Military service termination.
- Change of percentage of recompletion of the unit.
- Declaration of general mobilization.
- Recompletion of units during war.

The personnel supply is done during war with reserve or with active duty personnel (if available).

Consumer Material Supplies.

This type of supply includes all types of materials needed for the unit to fulfill its mission successfully. That includes ammunition, food, clothes, fuel and lubricant, medical supply, parts, and other needed materials.

Facilities.

This includes planning, maintenance and inspection for

all needed facilities to the armed force units. Some civilian constructions and facilities may be mobilized during war time according to facilities' mobilization plan. For example, some civilian hospitals may be dedicated to the military, some civilian factories and workshops maybe dedicated to the military also.

Transportation.

This includes transportation of material, personnel, and supplies to and from military units during peace and war time. This may be done by the unit and formation transportation means or by other means according to the transportation plan.

III. The Integrated Information System

The integrated information system of the Egyptian Armed Forces consists of two main systems as described in Figure 1.

- Logistic Support Systems
- Command and Control System

Logistic Support Systems

The term logistic is used in both military and industrial fields. From the military point of view Webster defines logistic as:

> The procurement, maintenance, and transportation of military material, facilities, and personnel. (1:4)

A United States Air Force technical report defines logistics as:

> The science of planning and carrying out the movement and maintenance of forces. In its most comprehensive sense, logistics pertains to those aspects of military operations which deal with:
> a) design and development, acquisition, storage, movement, distribution, maintenance, evacuation, and disposition of material,
> b) movement, evacuation, and hospitalization of personnel,
> c) acquisition or construction, maintenance, operation and disposition of facilities,
> d) acquisition or furnishing of services. (1:4)

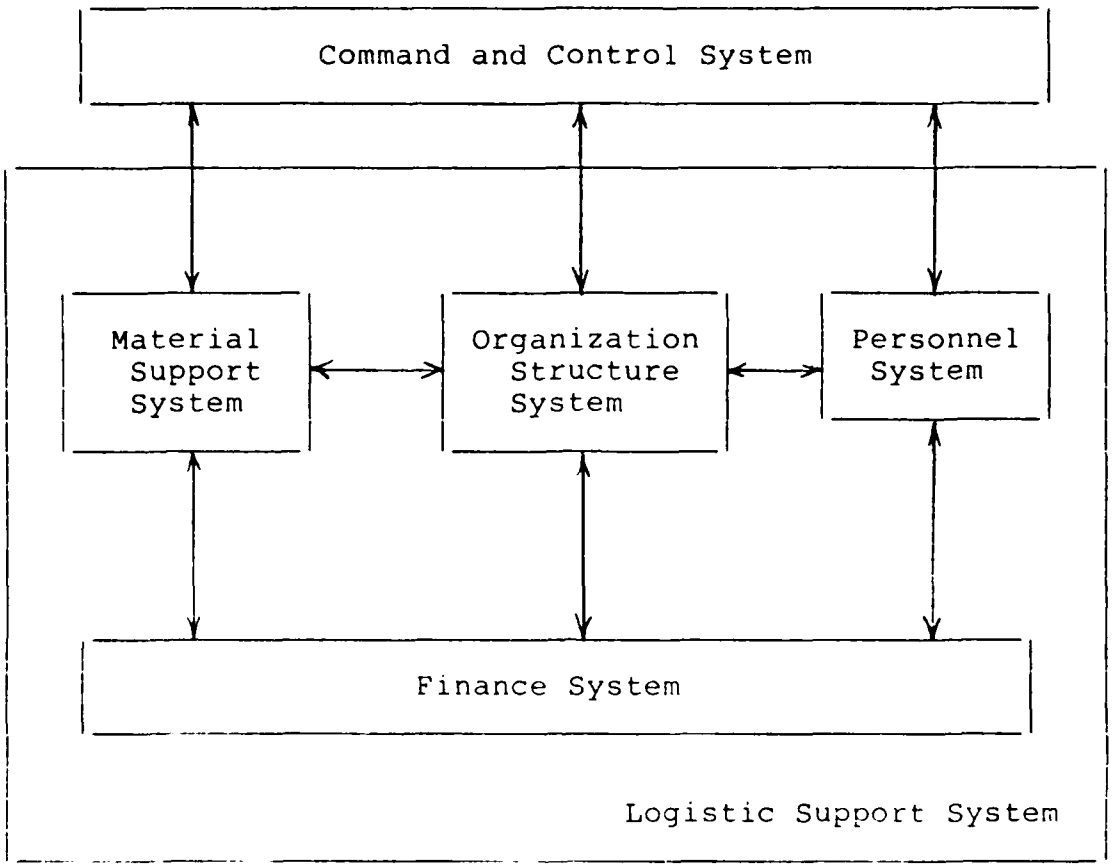


Figure 1. Integrated Information System for the Egyptian Armed Forces

In that sense, the logistic support system should include the elements of planning, acquisition, maintenance, test and support equipment, supply support, transportation and handling, personnel management, and all needed management and technical data. The main purpose of the logistic supply system is to supply personnel, material, facilities and technical support to the armed force units and formations at the right time with the right quantity.

The management of the organization structure of the armed force is one of the most important activities in the logistic support system. That is the maintenance and support of the organization structure of the armed force according to the national defense plan and available defense budget.

The integrated logistic support system will contain four main systems:

- Personnel Systems
- Organization Structure System
- Material Support System
- Finance System

Personnel Management System.

The personnel system should manage the information about the following:

- Officers
- Enlisted Personnel
- Draftees
- Reserve
- Civilians

There is not a big difference between the design and the needed information for both officers and enlisted sub-system, because both are professionals and usually spend most of their productive life in the military service. Both sub-systems should keep track of all necessary information needed for promotion plan, reassignment plan, training plan, and retirement plan besides necessary personnel records. The recruiting sub-system is not so simple. It should keep track of the personal information during three phases of the person's life:

- Before military service as a civilian
- During military service as a draftee
- After military service as a reserve

Before Military Service.

Every Egyptian male must have an ID card at the age of 16 years old by the law. The personnel data are transferred from the civil recording offices to the recruiting department which maintain that record and amends it if necessary according to education and health records. This data will be processed to prepare the recruiting plan according to the armed force's needs.

During the Military Service.

The system should be able to select personnel to certain jobs in certain departments according to the needs of the armed force units. The training plan for the draftees should be done for each department. Because there is a different training period needed for each type of personnel specification, the system should be able to keep track of that information and prepare the assignment plan, then follow up the execution of this plan.

The system should keep track of the personnel records during the military service (promotion, new assignment, punishments, etc.). The military service termination plan should be prepared according to the regulation stated by the law.

After the Military Service.

Every draftee must stay a certain period of time in the reserve according to the law, except some special cases stated by the law. The person should know the newly assigned unit in the reserve service (usually it will be the same unit) and how to reach it.

The mobilization may be done for training purposes or for war. The system should be able to keep track of all reserve information and prepare mobilization plan for both peace and war time and also follow-up its execution.

The civilian sub-system is a typical personnel system. It should keep track of personnel records, prepare reassignment, promotion, and retirement plans. The system also should estimate the real needs from the civilian specifications.

Management of the Organization Structure of the Eyptian Armed Forces

Establishment of a military unit is done through an organized process according to the real needs of the armed force to fulfill the national defense plan, putting in consideration the availability of the following:

1. Arms and Equipments
2. Manpower
3. Facilities
4. Supply
5. Budget

The unit is considered to be established as soon as its organization instruction is issued. The organization instruction is regarded as the "birth certificate" of the unit. A copy is sent to main [illegible] and departments.

Each organization instruction has a unique serial number. Many units may be established by the same organization instruction. Any change in the unit establishment data must be done through another organization instruction.

The organization instruction specifies the sponsor department and lists the departments responsible for supplying the newly established unit with personnel, arms, equipment, and facilities.

Each department mentioned in the organization instruction must assign the necessary personnel, arms, equipment, and/or facilities by the due dates stated in the instructions. The organization instruction should contain the following information:

1. General description of the unit
2. Description of the unit internal structure
3. Description of logistic support
4. Full description of personnel needed for the unit establishment according to the unit internal structure (this is summarized in the organization table).
5. Full description of arms and equipments needed for the unit established according to the unit internal structure (summarized in the armament table).
6. Timetable for establishment phases.

The following unit data must be stated in the organization instruction:

1. Unit name - unique name to identify the unit.
2. Main branch - army, navy, air force, air defense, or border guard.
3. Sponsor department - armor, signal corp, artillery, etc.
4. Unit level - division, brigade, battalion, etc.

5. Unit type - This attribute (with department and unit level) identifies the unit type. For example, if (3) and (4) describe an infantry battallion, (5) will describe the infantry battalion as mechanized infantry battalion or airborn infantry battalion or other type of infantry battalion.

6. Unit number - this number is a unique number with respect to each unit type to differentiate between different units of the same type.

7. Percentage of Completeness - According to the primary mission of the unit, the percentage of completeness of personnel may be identified to be less than or equal to 100%. The unit may be completed by mobilization for training reasons or for war reasons. New units are established according to the available arms and equipment. For tactical reasons, arms and equipment should be completed even if the completeness of personnel is less than 100%.

8. This attribute describes the unit's main missio from the point of view of armed force general balance. It may be one of the following six types:

   a. Command and control (e.g.) signal corp units.

   b. Fighting (e.g.) infantry or tank battalion

   c. Fighting assistance (e.g.) engineering battalion.

   d. Logistic (e.g.) transportation battalion.

   e. Technical (e.g.) armor field workshop.

   f. Training (e.g.) artillery school.

9. Location - The organizational instruction must identify the primary unit location. Egypt is divided into six main military location areas.

10. Command Unit - Each unit must be commanded by only one unit (except the MOD - that is regarded as the root of the armed forces structure tree).

11. Commanded Sub Unit(s) - The organization structure should state other units commanded by the established unit (if any).

12. Defense Plan - This attribute defines which defense plan this unit has a mission. The unit may be assigned to more than one plan.

13. Personnel Table - This table summarizes the required personnel for different departments according to the five main specifications. Figure 2 illustrates an example of a personnel table.

14. Armament Table - This table specifies the required arms and equipment needed for the unit (types and quantities). The department that is to supply that type of equipment may not be stated.

The armed force instructions define which departments supply which arms/equipment. In some cases, the department may be stated to eliminate ambiguity if necessary. Figure 3 illustrates an example of an armament table. Notice that only the main arms and equipment are listed in the table. If the newly established unit has the same organization as an existing unit, the organization instruction refers the organization instruction number of the existing unit and both organization and armament table may not be repeated.

User Needs from the Organization Structure Information System.

1. Keep track of the 14 attributes mentioned in the organization instruction for each unit.

2. Define the relations between the different types of units.

3. Create the hierarchical information model for the armed force units.

Type / Dep.	Officer	Secretary	Commission	Driver	Technician	Craftee	Total
Infantry	16	4	320			1	341
Armor	1			25	6		32
Signal Corp	1		7		2		10
Chemical WF	1		4				5
Wheeled Veh.				12			12
Reconnais-sance	1		6				7
Medical Corp.	1			4			5
Total	21	4	337	41	8	1	412

Figure 2. Personnel Table for a Mechanized Infantry Battalion (Example)

Small Arms

9 mm pistol	64
Automatic gun	336
Machine Gun	9
RB J-7	9

Artillery

82 mm mortar	4
Guided anti tank missile (tow)	6
Anti Air Craft Guided Missile (SAM-7)	3

Armored Carrier

M 113 - A2	24
M 113 - Recovery	1

Wheeled Vehicles

4 x 4 Jeep 1/2 Ton	6
4 x 4 Truck 1 1/2 Ton	2
4 x 4 Truck 5 Tons	10
Fuel Truck	2
Water Truck	1
Ambulance	2
Maintenance Truck (Armor Dept.)	1

Other Equipment

Generator 10 kw	1
Wireless Set R124	4
Wireless Set R127	1

Figure 3. Armament Table for a Mechanized Infantry Battalion (Example)

4. Be able to answer queries like:

   a. Describe the organization tree for the 10th Armor Division and total number of personnel for each unit in the tree and for the division.

   b. Identify the types and number of personnel needed from the signal corp department to supply the units sponsored by the armor department.

   c. Identify the personnel requirement to the fifth field army by type and number from the specialized departments.

   d. How many Howitzer 160 mm or 240 mm should exist in the west location?

   e. What armor unit exists in Central location to support plan A?

5. Identify and secure the interface between the system and the other systems (personnel, finance, and material support) and with command and control system.

Material Support System

This system should perform the procuring, maintaining, and securing of the following:

1. Complete Arms and Equipment
2. Spare Parts
3. Daily Consumed Material (food, fuel, clothes, medical support, etc.).
4. Ammunition
5. Facilities
6. Transportation
7. Other Military Material

This system deals with a wide variety of military

material. But fortunately, most of it is related. For example, one of the system's main jobs is related to the military arms and equipments. The operation of A & E needs ammunition and fuel, the technical support of A & E needs parts and lubricants and both of them need transportation, so most of the data related to the A & E is still in the same system. On the other hand, the system still needs some information from the organization structure system about unit structure, location, and armament table for each unit.

Another type of information needed from the command and control system is the operational mission for units to be able to secure suitable amounts of ammunitions, fuel, transportation, food, and water, procurement and maintenance of the material. This needs complete interaction with the finance system.

Finance System

No one can ignore the impact of the financial matter on all segments of the armed forces. Armament, personnel, supply, and facilities are examples of those segments that are highly affected by financial decisions.

The function of the finance system is defined by Herbert T. Spiro as:

> Finance engages in two primary types of functions for top management: recording, monitoring, and controlling of financial consequences of past and current operations, and securing funds to meet current and future needs. (5:1)

To accomplish the first set of functions, the finance system needs well-defined interface with material support, personnel, organization structure and command and control systems. The second set of functions needs interfacing with outside institutions (e.g., Ministry of Treasury, Ministry of National Planning, etc.). Discussion of the finance system is out of the scope of this project.

Command and Control System

To describe the system of Command and Control (C^2), note Dr. Robert E. Conley's definition:

> Command and Control (C2) is a process of resource allocation (management) by a knowledgeable, recognized point of authority to accomplish a given objective. (3:16)

This definition describes the C^2 system as the process by which the military objective(s) may be accomplished with the available resources. Actually, the definition is applicable for military and other management areas and the authority may not be a human. The C^2 system cannot increase the resource capability but it helps to allocate the resource at the appropriate time and place. Generalization of the C^2 system results in the "command, control, communication and intelligence" system given a scope of information that should be collected and processed through this system. Two categories of information should be

manipulated by the system; our own forces and the opposing forces. Types of information and functions of the C^2 system are described by the official definition of command and control in the Joint Chief of Staff (JCS) Dictionary as:

> The exercise of authority and direction by a properly designated commander over assigned forces in the accomplishment of his mission. Command and Control functions are performed through an arrangement of personnel, equipment, communication, facilities, and procedures which are employed by a commander in planning, directing, coordinating, and controlling forces and operations in the accomplishment of his mission. (3:26)

The components of the C^2 system are defined by J. S. Lawson as:

1. Sensors
2. Communications
3. Data Processing
4. Information Management
5. Decision Aids
6. Forces to Command
7. A Commander (3:62)

A basic model of C^2 process is described in Figure 4 (3:66).

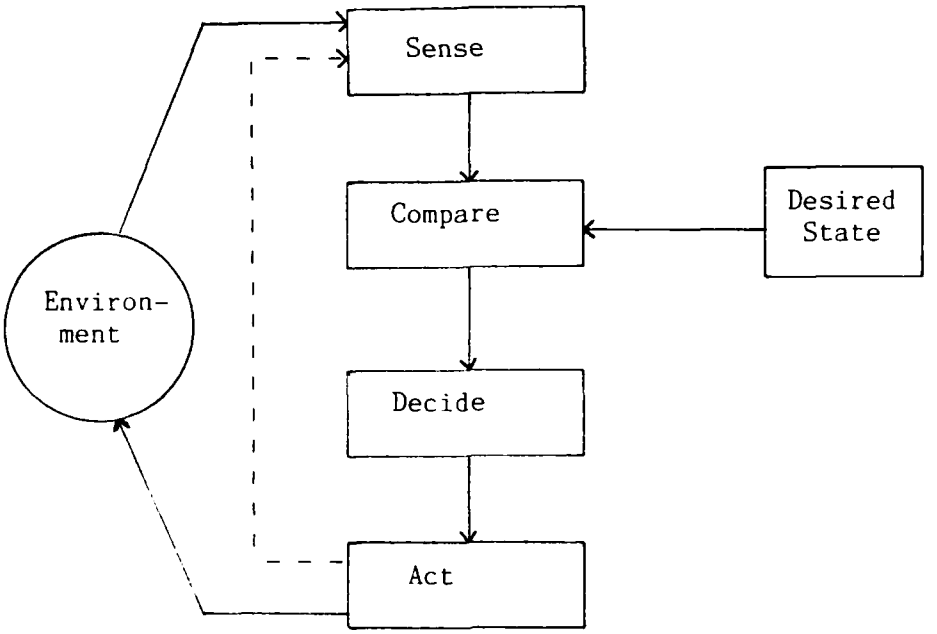


Figure 4. Basic Model of C^2 Process

Scope of the Project

From the previous discussion about the Egyptian armed forces integrated information system, we can conclude that the first step should be the design of the organization structure system. All of the systems included in the integrated information system need information about the organization structure of the units and formations and/or information related to the organization table and/or armament table for units.

The organization structure system may be designed and implemented by itself because the original source of data is the Organization Instruction for units. The percentage of completeness, location, and defense plan may be changed by operational instruction. Later, this may be done through the interface with the command and control system. For those reasons, the scope of the project will be the design of the organization structure information system using a data base management system. The implementation will be done as time permits. The designed model should define and manipulate the following data which is described earlier in this paper.

1. Organization Instruction Number
2. Unit Name
3. Main Branch
4. Sponsor Department
5. Unit Level
6. Unit Type
7. Unit Number
8. Percentage of Completeness
9. Balance Type
10. Location
11. Commant Unit
12. Commanded Sub-Unit(s)
13. Defense Plan

14. Organization Table

15. Armament Table

The thesis work will cover the following areas:

- Definition of the System E-R Model
- Definition of the System F-D Model
- Choice of DBMS Model
- Normalization (if necessary)
- Design of Conceptual Data
- Implementation (as possible)
- System Recovery and Concurrency
- System Security and Integrity
- System Interfaces

This thesis work will cover the current information about the organizational structure of several Egyptian military organizations. The information is needed for planning and management of the armed forces organization structure. The implementation of this project keeps the consistancy and correctness of data over all of the armed forces organization.

IV. Design of Entity Relationship Model

The E-R model is one of the most useful tools to summarize the designed data base information that may be represented as three different types:

Entity

> The term "entity" is widely used in data base circles to mean any distinguishable object that is to be represented in the data base. (2:10)

A group of similar entities may form an entity set. The entity will be represented by rectangles.

Attribute

> Entities have properties called attributes, which associate a value from a domain of values for that attribute with each entity in an entity set. (8:6)

The attributes' definition for each entity is an important step in data base design. If an attribute or a set of attributes whose values uniquely identify each entity in an entity set is called a key attribute(s), the attributes will be represented by circles (or ovals) and the key attribute will be marked by an asterisk.

Relationship

> A relationship among entity sets is simply an ordered list of entity sets. A particular entity set may appear more than once on the list. If there is a relationship REL among entity sets E_1, E_2, ..., E_K, then it is presumed

that a set of k tuples named REL exists. We call such a set a relationship set. Each k-tuple (e_1, e_2, ..., e_k in set REL implies that e_1, e_2, ..., e_k, where e_1 is in set E_1, e_2 is in set E_2, and so on, stand in relationship REL to each other as a group. (8:7)

The relationship will be represented by diamonds.

Entity Types

Figure 5 illustrates the designed E-R model for the system. The newly established unit may follow one of the existing models (as described in the last section), or it may be established by a new unit model. For example, all M60-43 tank battalions have the same structure (i.e., they follow the same unit model). Five entity types are defined; the first one is "Unit-Model" which represents a prescribed structure of units that may be followed by a newly established unit; the second one is "Unit" which represents an existing real unit. Two entity types "Person" and "Armament" describe both personnel and armaments assigned to the predefined unit-model. The last entity type is "Plan" which describes assigned plan for unit (See Figure 5).

Attribute Types

Unit Model Attributes.

There are three attributes (udep, level, type) that uniquely identify the unit-model and another three attributes.

udep - unit sponsor department number. It is a 2 - digit integer number ranging from 1 to 99.

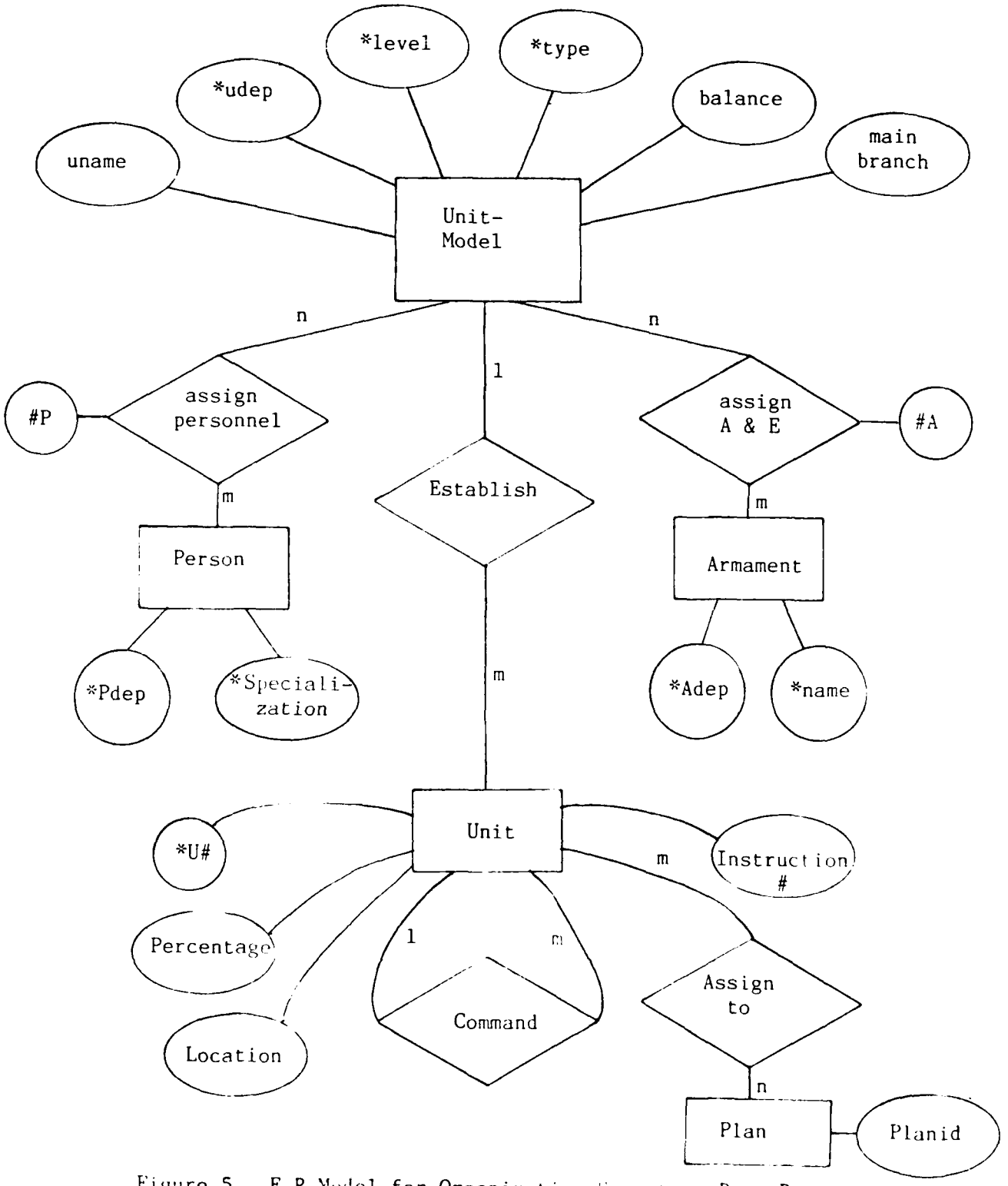


Figure 5. E-R Model for Organization Structure Data Base

*Level - Unit level (division, brigade, battalion, etc.). It is a 2-digit integer number ranging from 00 to 99 (level 00 represents MOD only).

*Type - Unit type. If the unit is infantry battalion, this attribute describes its type (mechanized, air born, amphibious, etc.). It is a 2-digit integer number ranging from 00 - 99. (Type 00 means that this unit has only one type, e.g. MOD, Infantry Department.)

Uname - Unit name like "mechanized infantry brigade". Unit name may identify the unit model and it is regarded as candidate key (a relation's attribute that uniquely identifies all relation's attributes is called a candidate key). We must add unit number (the key attribute of unit entity) to Uname to be "Mechanized Infantry brigade Number 115". It is represented by a string of 30 characters.

Balance - It represents the unit balance type (command & control): fighting, fighting assistance, logistic, technical, and training. It is represented by an integer number ranging from 1 - 6.

Main Branch - The main branch which the unit model belongs to (army, navy, air force, air defense, border guard). Its domain is the set of values (AR, NA, AF, AD, BG).

Person Attributes.

Person entity represents the organization table as described in Figure 2. There are two key attributes and another attribute.

*Pdep - Person department number. It is the specialized department responsible for supplying the unit with the defined type of personnel. It is a 2-digit number ranging from 01 - 99. [illegible] same attribute value as Ulev.

*Specialization - It is defined as the person specialization and it is one out of six specializations (officer, secretary, commission, driver, technician, or craftee). Its domain value is the set (off, Sec, Com, Dri, Tec, Cra).

#P - It is the attribute of the relationship assigned personnel and it is the number of personnel from the predefined Pdep and specialization. It is an integer number ranging from 0 - 999.

Armament Attributes.

Armament entity represents the armament table as described in Figure 3. There are two key attributes and another one.

*Adep - Armament department number. It is the specialized department responsible for supplying the unit with the defined type of arms or equipment. It is a 2-digit number ranging from 01 - 99. (the same attribute domain as Udep.)

*name - Arm or equipment name. It is represented by 30 Character String.

#A - It is the attribute of the relationship assign A & E and it represents the number of arms or equipments predefined by both Adep and name. It is an integer number rangingfrom 0 - 999. (The same attribute value as #P.)

Unit Attributes.

There is only one key attribute (U#). This attribute does not uniquely identify a unit. Unit is identified by unit-model keys (udep, level, type) and U#.

*U# - unit number. This attribute defines the unit number for an existing (or new established) unit which has a unit model existing in the entity set Unit-Model.

If the unit model can represent only one unit (e.g.) unit model of MOD or logistic authority, the unit number will be zero, Otherwise, the unit number will be greater than zero. Units of different unit model may have the same U#. U# is represented by an integer ranging from 0 - 999.

Percentage - percentage of completeness of the unit from personnel (in peace time). It is an integer ranging from 0 - 100. (As a matter of fact there is no zero percentage of completeness as it means no personnel in this unit but it may be considered for special cases of reserve units where the whole unit is stored.)

Location - The geographical location of the unit. Egypt is divided into six military locations (A, B, C, D, E, F). Its domain value is the set (A, B, C, D, E, F).

Instruction # - The organization instruction number that establishes the unit. It is represented by an integer number ranging from 1 to 99999.

Plan Attribute.

Plnid - The defense plan(s) which the unit has a mission with. The unit may be assigned to more than one defense plan and more than one unit may have the same plan (min). Its domain value is the set of Alphabet (A, B, C, ..., Z).

Relationships

First, relationship "established" between Unit-Model and Unit represents that each unit is established within a unit model. As many units may have the same model, the relationship is a one to many relationship. Second and third relationships "assign personnel" and "assign A & E"

represent the relationship between unit model and personnel and armaments assigned to that unit model consequently. Both relationships are many to many (more than one occurrence of the first entity may be assigned to one occurrence of the second entity, and more than one occurrence of the second entity may be assigned to one occurrence of the first entity). Different personnel from different departments and different specializations may be assigned to the same unit-model, and personnel from the same department and from the same specialization may be assigned to different unit models.

The fourth relationship "command" represents the hierarchical relationship between units. The command unit (which is unit) commands a set of units and each unit (except MOD which represent the rest of the tree) is commanded by only one unit. Therefore, all units (except the root) are involved in this relationship. The other option to represent the "command" relationship is to consider the commanded units. From this point of view we can see that all units have commanded units except "leaf-units". The number of units involved in the second option of the relationship is much fewer than first option; therefore, we will choose the second option. The last relationship is assign-to describes which unit is assigned to which plan(s). This relationship is a many to many relationship.

Function Dependency

The entity Unit-Model has three key attributes (udep, level, and type). These three key attributes define uname and balance as described in Figure 6.a. The other attribute "main branch" can be defined by udep only.

The relationship "assign personnel" is a many to many relationship between unit-model and personalities; #P can be defined by means of the key attributes of both entities (udep, level, type, Pdep, and specialization) as described in Figure 6.b.

A similar functional dependency exists between Unit-Model and armament entities; #A can be defined by means of the key attributes of both entities (udep, level, type, Adep, and name) as described in Figure 6.c. The unit attributes percentage, location, and instruction #, are defined by Unit Model's key attributes Udep, level, and type and Unit's key attribute U# (see Figure 6.d).

The last relationship "command" describes the command unit for each unit. Therefore, the command unit key attributes cudep, clevel, ctype and cu# (c stands for command unit just to eliminate ambiguity) are defined by unit attributes udep, level, type, and U# (see Figure 6.e).

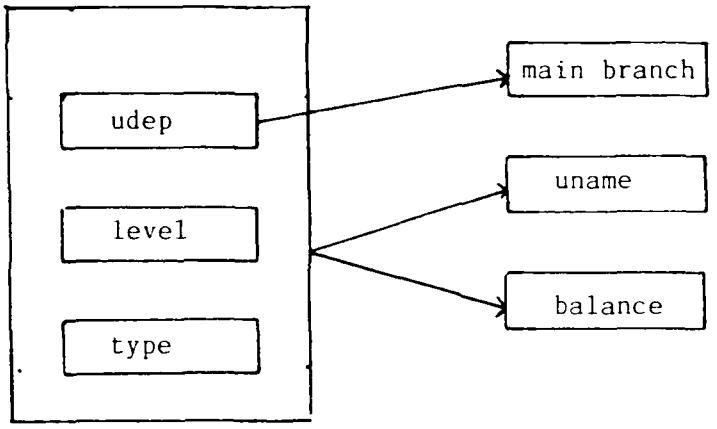


Figure 6-a. Function Dependencies for Unit-Model Entity

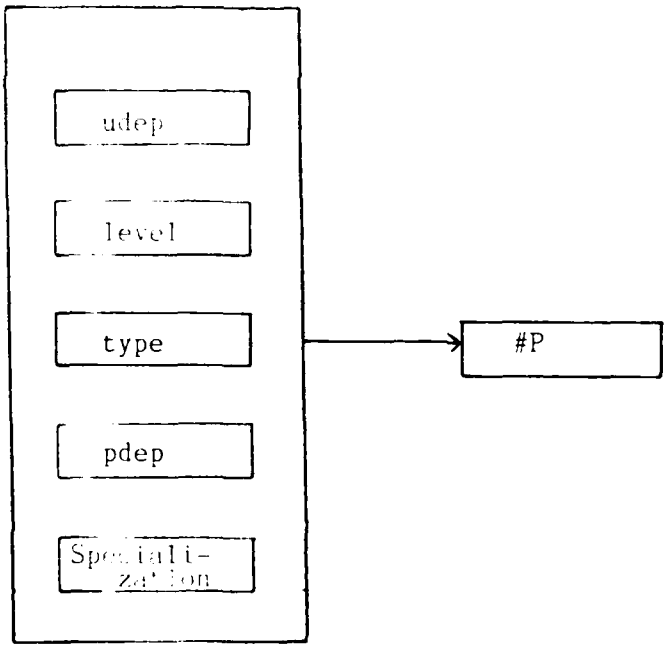


Figure 6-b. Function Dependencies for Assign-Personnel Relationship

Figure 6. Function Dependencies Diagram

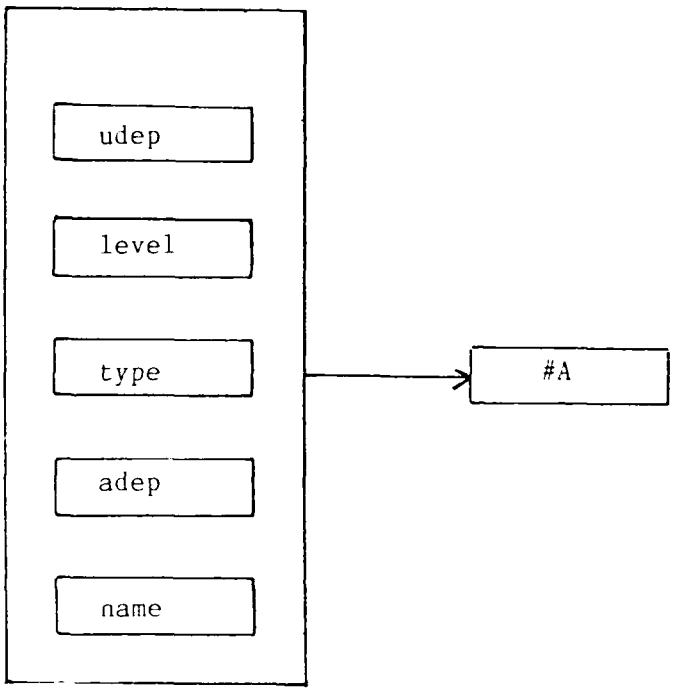


Figure 6-c. Function Dependencies for Assign A & E Relationship

Figure 6. Function Dependencies Diagram (Cont.)

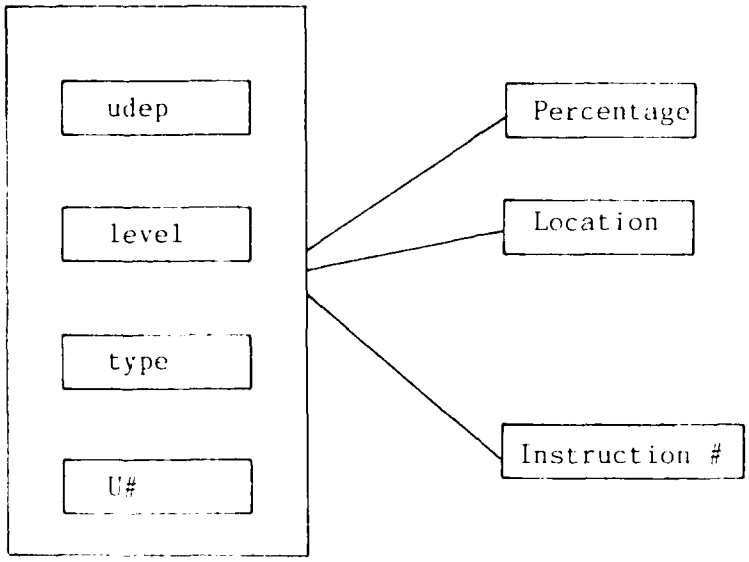


Figure 6-d. Function Dependencies for Unit Entity

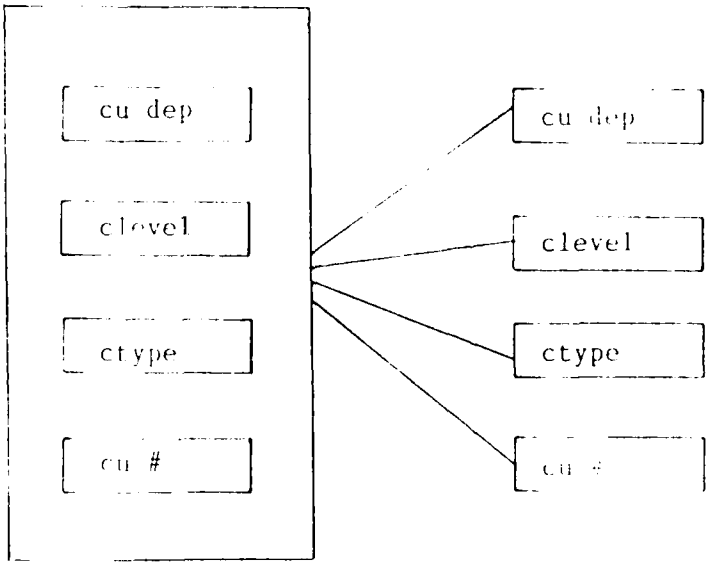


Figure 6-e. Function Dependencies for Command Relationship

Figure 6. Function Dependencies Diagram (Cont.)

V. Design of Data Base

Choice of DBMS Model

The process of choosing a DBMS Model for implementation needs to compare the feature of the three main DBMS models; hierarchical, network, and relational models which will be represented by DL/1, IDMS, and SQL, respectively. This comparison will be done through a framework for the evaluation of data model which is described by Vasta (9:219-226). Figure 7 summarizes this framework of evaluation (9:220).

General Considerations.

Data Building Blocks.

It is the smallest unit of data. It is defined as a field in DL/1, data item in IDMS, and attribute in SQL. The data transfer unit is a segment in DL/1, and a Record in IDMS. The relational model has an advantage over the two other models; it offers two units of data transfer. SQL permits transfer of one tuple at a time, and when data is operated upon through an on-line terminal in SQL, the units of data transfer is a table (8:221).

Representation of Logical Data Structure.

The logical data structure of the design model as described by E-R diagram (Figure 5) represents two one to many relationships and three many to many relationships.

	HIERARCHICAL	NETWORK	RELATIONAL
Representative system	DL/I (IBM)	IDMS (Cullinet	SQL—the DML for IBM's DATABASE 2
Data building blocks	Field Segment Physical data base	Data item Record Set	Attribute (column) Tuple (row) Relation (table)
Representation of logical data structures			
—*Trees*	Directly	Decomposition into sets	Decomposition into tables
—*Simple networks*	Unidirectional logical relationship	Decomposition into sets	Decomposition into tables
—*Complex networks*	Bidirectional logical relationship	Decomposition into sets using intersection data	Decomposition into tables using intersection data
Data independence			
—*Path*	No	No	Yes
—*Sequence*	No	No	Yes
DML commands			
—*Retrieval*	GU, GHU, GN, GHN, GNP, GHNP	FIND, GET	SELECT
—*Data alteration*	REPL	MODIFY	UPDATE
—*Data addition*	ISRT	STORE	INSERT
—*Data deletion*	DLET	ERASE	DELETE
—*Miscellaneous*		READY, FINISH, CONNECT, DISCONNECT	
Means of data base navigation	Through hierarchical path	Through sets	Through the value of the attributes
Navigator	Experienced trained programmer	Experienced trained programmer	End user
Performance	High—with well-defined access paths Low—with unstructured access paths	High—with well-defined access paths Low—with unstructured access paths	High—with unstructured access paths Low (in comparison to hierarchical and network data models)—with well-defined access paths
Additional pointers available to improve performance	Yes—retrieval	Yes—retrieval	Yes—but SQL may choose not to use them
Security	Defined in subschema	Defined in subschema	Defined in subschema—but may be modified at any time including during on-line execution
Security officer	DBA or equivalent	DBA or equivalent	DBA or equivalent—may be delegated
Responsibility for adding data to the data base	Data base management system—governed by INSERT, REPLACE, DELETE rules	Data base management system or application program—governed by insertion status	Data base management system
Modification of the data structure	Redefine structure, reload new structure	Redefine structure, reload new structure	Restructure at any time, including operation in an on-line environment

Figure 7. Framework for the Evaluation of Data Model (9:220)

The logical data structure may be represented as a network. The network structure cannot be represented directly by DL/I. Data is represented as if it were duplicated. Data are reproduced in two different segments in a tree structure then logical pointers are used to eliminate redundancy. Network model represents many to many relationships by creating a record type which contain intersection data for records involved in the many to many relationship. The intersection record is a member in each of two sets in which the related records are owners (8:221). Relational model also creates an intersection relation including primary keys of the relations involved in the many-to-many relationship.

Data Independence.

DL/I allows logical sequencing of segments and permits secondary indices. While this provides great flexibility to the application program, it causes sequence dependency and also path dependency since the desired segments is obtained through a specific path. IDMS is also subject to the same sequence and path dependency.

In relational model, pointers are not used and the relationship is kept through the tables' attributes. In a relational model, columns and rows may be presented in any order (i.e., there is no specific sequence that must be followed). Therefore, the relational model is free from both sequence and path dependency.

Navigator.

Navigation is done in both hierarchical and network models through specific paths. In a hierarchial model, it is done through the traversal of the tree structure and in a network model, through the traversal of owner and member sets. In both models, the navigator must know the current location in the data base and the direction of the desired data item. Therefore, a highly trained programmer is needed.

Because of the data independence in the relational model, there is no specific path to follow, and the data is obtained by defining the value of the attributes within relations. It is not necessary to keep track of the current data base position, or to describe how to find the data. Therefore, there is no need for a highly trained programmer. The end user with little training can manipulate data.

Performance.

In a hierarchical model, the specific paths which are defined by pointers permit rapid retrieval of desired data. The same situation is also in a network model. On the other hand, "neither IDMS nor DL/I is well suited to an unstructured environment where data relationships can be changed while the user is accessing the data base" (8:223).

The relational model needs greater space than both the hierarchical and network models. SQL is more powerful as it performs operations on many tuples within a single query. Where DL/I and IDMS operate on one record or segment at a time and may require many commands to accomplish one SQL command.

Security.

In a DBMS, security is defined on the level of the subschema. In both DL/I and IDMS, the subschema definition is translated before execution. In SQL, the definition of the subschema may be modified at any time. (8:224)

SQL has additional security tools; that is "Grant and Revoke" mechanism and "view" mechanism. The view mechanism allows the data to be conceptually divided up in such a way to that some data may be hidden from unauthorized users. The Grant and Revoke mechanism or the authorization subsystem lets the authorized user grant rights to any other user(s) but not exceeding his own rights, and also Revoke granting rights to the user he granted and any other user that granted user rights (2:441-443).

Modification of the Data Structure.

Since both hierarchical and network models are data dependent and the data are linked together in such a way that it is needed to follow a certain path to retrieve a specific segment, modification of data structure needs a

working copy of the data base(s) to be made, a new definition of the data structure, and then recreate the data base in the new structure from the working copy. While the restructuring is occurring, programs may not access or modify the existing data (9:224). In case of relational model, the DBMS is data independent, so the data structure may be modified at any time while the system is running and without recreation of the data base.

Comparison of Models.

From the above discussion and considering the shape of the data base that is defined by the E-R model described in Figure 5, we can summarize our data base features as:

1. There are two one-to-many relationships and three many to many relationships.
2. There must be an algorithm to create the formation hierarchical relation (the relation that contains the set of units which are commanded by a particular unit) from the command relation (the relation that defined the subunits (if exist) for each unit).
3. We should consider that the data base will have many interfaces to other systems (see Figure 1) and the homogenious aspects of DBMS for the armed force integrated system.

From the above features of the data base we will exclude the hierarchical model from our consideration for the following main reasons:

1. There are three many to many relationships.

2. Hierarchical model cannot represent a network directly; instead, it's decomposed into many tree structures that may cause data redundancy.

Now the comparison will be done between the network model and the relational model.

Network Model.

Advantages:

1. Simple representation of network
2. Rapid Response

Disadvantages:

1. Data dependent
2. Needs experienced programmer
3. Modification of data structure needs to redefine structure and related new structure.

Relational Model.

Advantages:

1. The data transfer unit may be a tuple or a whole table.
2. Data independent.
3. Simple data manipulation, that end user with little training can manipulate the data.
4. Additional security tools (view, Grant-Revoke).
5. Easy modification of data structure.

Disadvantages:

1. Needs greater space.

2. Slower response.

Choice Considerations.

From the above discussion and from the advantages and disadvantages of network and relational model, consider the following:

1. The designed data base will have interfaces to other systems that most likely to use rational models.

2. "Networks are complex and the structure is complex. This increase in complexity compared with relational system does not lead to any corresponding increase in functionality" (2:574).

3. To keep the integrated system as a homogenous system, we should use the same DBMS model.

4. Ability to have relational model for the Egyptian VAX systems.

From the above, the chosen data base model is the relational model and the implementation will be done within DBMS on a VAX 780 (SSC system).

Design of Logical Data

In a relational data base model, the data base is arranged into two dimension tables (relations). Each occurrance of data (row) is called a tuple. A tuple consists of one or more properties called attributes. Each

attribute can take on a specific set of values over a period of time. All the values that an attribute can take on are called the domain of the attribute. (9:198-199).

Retrieving the E-R model described in Figure 5, we will start defining the data base relations as a relation for each entity and for each attribute. The second step is to use a computer normalization tool to normalize any non-normalized relation to third normal form (if any).

Direct Relations Definition.

The direct relations definition for the data base described by the E-R diagram (Figure 5) will be described below. The relation name will precede the relation. The "*" preceding an attribute represents it as a key attribute:

1. Unit Model
   * dep
   * level
   * type
     name
     balance
     main branch
2. Person
   * p[illegible]
   * specialization

3. Armament

   * Adep

   * name

4. Assign Personnel

   * dep

   * level

   * type

   * Pdep

   * specialization

     # P

5. Assign - A & E

   * dep

   * level

   * type

   * Adep

   * name

     # A

6. Unit

   * dep

   * level

   * type

   * U#

     Percentage

     Location

     Instruction #

7. Plan

   * dep
   * level
   * type
   * U#
   * Plan

8. Command

   * c dep
   * c level
   * c type
   * c U#
   * dep
   * level
   * type
   * U#

Normalization.

The purpose of normalization is to design data structures in such a manner as to eliminate the need to redesign them when new applications are added. By storing data in the third normal form, structures designed today can be integrated with new application in the future with little, if any, redesign effort (9:254-255).

Using a CAD normalization tool which is running on LSI-11 System "A" by Mallory (4), we found that the two relations that need further normalization. The first relation Unit-Model is normalized as the following two relations:

1. Unit-Model.

   * dep

   * level

   * type

     name

     balance

2. Department.

   * dep

     main branch

The plan relation is normalized to the following two relations:

1. PLAN.

   * dep

   * level

   * type

   * U#

   * plan

2. PLANID.

   * id

Data Definition

The data base of the organization structure of the Egyptian armed forces after the normalization step will be presented in ten relations. Referring to the attributes description presented previously and using ingres notations, the data base will be defined as the following set of relations:

1. Unit Model

Attribute Name	Type	Domain
* dep	C 2	digits 01:99
* level	C 2	digits 00:99
* type	C 2	digits 00:99
name	C 30	Alpha numeric
balance	C 1	digits 1:6

2. Department

Attribute Name	Type	Domain
* dep	C 2	digits 01:99
branch	C 2	(AR, NA, AF, AD, BG)

3. Person

Attribute Name	Type	Domain
P dep	C 2	digits 01:99
* Spec	C 3	(Off, Sec, Com, Dri, Tec, Cra)

4. Armament

Attribute Name	Type	Domain
A dep	C 2	Digits 01:99
Name	C 30	Alpha Numeric

5. AssignP

Attribute Name	Type	Domain
* [illegible]p	C 2	Digits 01:99
* level	C 2	Digits 00:99
* type	C 2	Digits 00:99
* Pdep	C 2	Digits 01:99
* Spec	C 3	(Off, Sec, Com, Dri, Tec, Cra)
NUMP	i 9	Integer 1:9999

6. Assign-A

Attribute Name	Type	Domain
* dep	C 2	Digits 01:99
* level	C 2	Digits 00:99
* type	C 2	Digits 00:99
* Adep	C 2	Digits 01:99
* Name	C 30	Alpha Numeric
num	i 3	Integer 1:999

7. Unit

Attribute Name	Type	Domain
* dep	C 2	Digits 01:99
* level	C 2	Digits 00:99
* type	C 2	Digits 00:99
* Num	C 3	Digits 000:999
Percentage	i 4	Digits 0:100
Location	C 1	(A,B,C,D,E,F)
Instruction	C 5	Digits 00001:99999

8. Plan

Attribute Name	Type	Domain
* dep	C 2	Digits 01:99
* level	C 2	Digits 00:99
* type	C 2	Digits 00:99
* Num	C 3	Digits 000:999
*Plan	C 1	(A, B, ..., Z)

9. Planid

Attribute Name	Type	Domain
* id	C 1	(A, B, ..., Z)

10. Command

Attribute Name	Type	Domain
* C dep	C 2	Digits 01:99
* C level	C 2	Digits 00:99
* C type	C 2	Digits 00:99
* C Num	C 3	Digits 000:999
* Dep	C 2	Digits 01:99
* Level	C 2	Digits 00:99
* Type	C 2	Digits 00:99
* Num	C 2	Digits 000:999

The new data base "OMADB" (OMADB stands for Organization and Management Authority Data Base) is created under UNIX operating system using ingress. OMADB consists of the ten relations UNITMODEL, DEPARTMENT, PERSON, ARMAMENT, ASSIGNP, ASSIGNA, UNIT, PLAN, PLANID, and COMMAND. Ingres data definition of the relations is presented in Appendix A.

VI. Data Manipulation

The data manipulation will be done through Ingres Data Manipulation Language (DML). Ingres uses English-like commands to manipulate data within relations. Each relation represents a table which is identified by name. Each row represents a tuple, each column represents an attribute of the relation which is identified by name. Ingres provides two sets of commands; the data definition commands that let the user define the needed information (ask questions), and data manipulation commands for insertion, updating, and deleting data. The user need not be aware of whether he uses base table or view to retrieve the required information (views represents the visible data for the user and it will be discussed later in Chapter VII). To represent data manipulation, let us start building the physical data base for the hypothetical formation presented in Figure 8.

Physical Data Base Representation

Figure 8 presents a hypothetical formation of units which consists of:

1. MOD which presents the root of the tree.
2. Three specialized departments; infantry, armor, and signal corp departments.

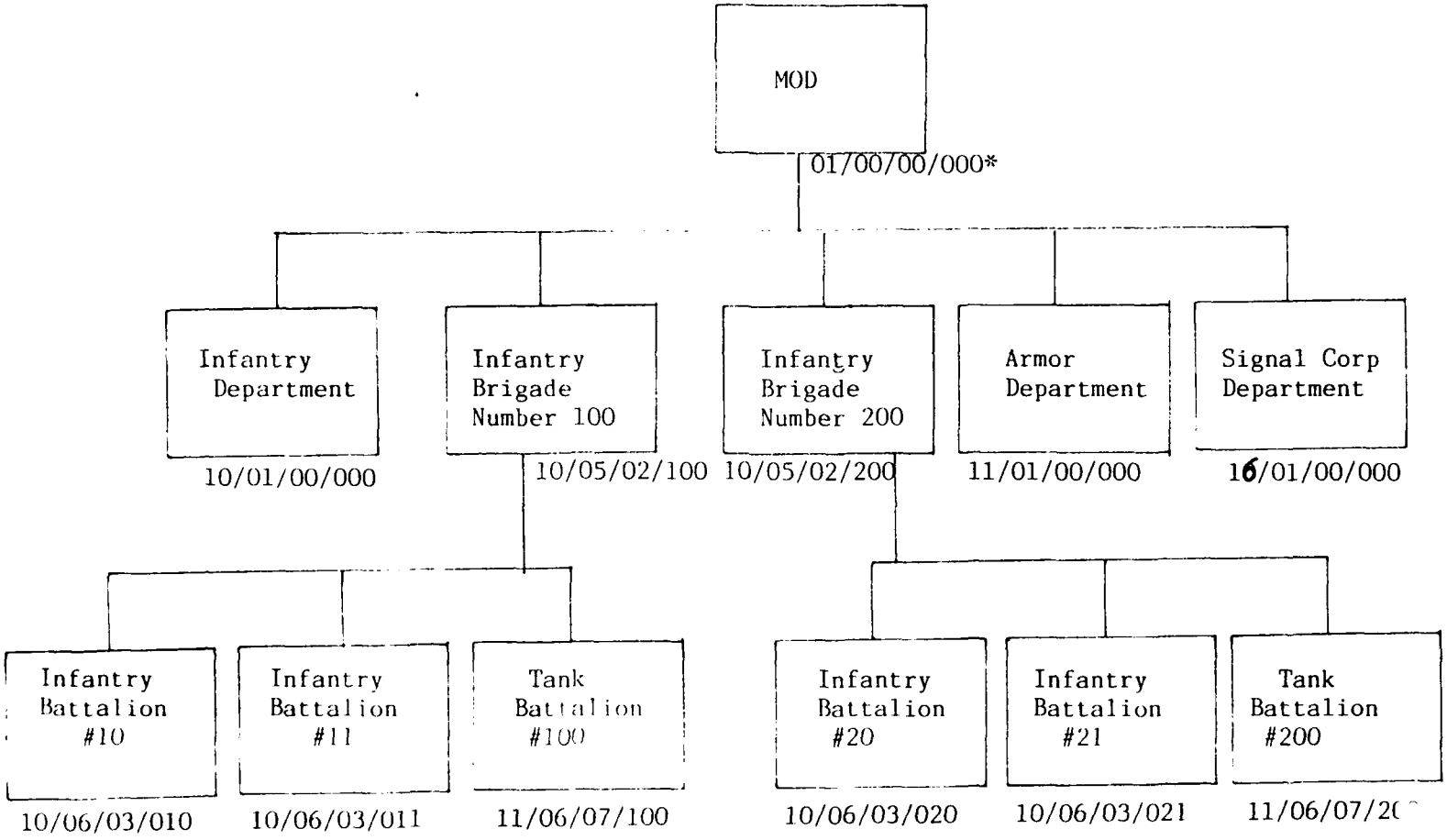


Figure 8. Unit Formation of OMADB Data Base

3. Two infantry brigades number 100 and 200 (both brigades have the same unit model).
4. Four infantry battalions number 10, 11, 20, and 21 (the four battalions have the same unit model).
5. Two tank battalions number 100 and 200 (both battalions have the same unit model).

Description of Physical Data Base.

All departments and MOD are located in Location A. Infantry brigade 100 and its subunits are located in Location B. Infantry Brigade 200 and its subunits are located in Location C.

All departments and MOD have balance Code 1. All other units (in the sample data) have balance Code 2. All units have main branch "AR" (army). Percentage of completeness for MOD is 100%, for departments is 70%, for brigade 100 and its subunits is 90%, and for brigade 200 and its subunits is 85%.

MOD is in plans A, B, and C. All departments are in plan A. Infantry brigade 100 in plans B, C. Infantry brigade 200 in Plan B except tank battalion 200 which is in plans B and C. Appendix I represents the personnel and armament tables for the sample units as described in the organization instructions. Representation of physical data of OMAR in the ten relations is presented in Appendix B.

Retrieval of Data

Retrieval of data in our model is done through the ingress data manipulation language that has the advantages of relational algebra in dealing with relations. Ingres deals with data as a set of data items (tuples) compared with the traditional methods of data manipulation that deal with each data item separately (record by record or segment by segment). In the particular case when ingress is looking for a particular tuple, it deals with it as if it's a set of data items consisting of one tuple. This new dimension in data manipulation which was introduced by relational algebra let the user write his query in more abstract form and make it easier for the inexperienced user to deal with the relational data base management system. This concept implies that dealing with data in sequential order (or in any other order) has no meaning. The reason for arranging the tuples in a particular relation is either to reduce the access time or a required output format. In the next section we will introduce some examples of questions that our system can support.

Examples of System Queries.

Ingres data manipulation language (DML) may be done interactively or imbedded in other languages like C.

The following are some examples of queries that are necessary for the management of the organization structure

of the Egyptian armed forces. These queries may be imbedded in other languages or simply as interactive queries.

a. How many infantry battalions exist in location b?

Ingres DML:

```
*range of u is unit
*retrieve into temp (u.num)
*where u.dep = "10" and u.level = "06"
*and u.type = "03" and u.location = "b"
*\g
*range of t is temp
*retrieve (number = count (tnum))
*\g
```

Result

number
2

b. How many officers exist in a tank battalion?

Ingres DML:

```
*range of p is assignp
*retrieve into temp (p.num)
*where p.dep = "11" and p.level = "06"
*and p.type = "07" and p.spec = "off"
*\g
*range of t is temp
```

*retrieve (nump = sum (t.num))

*\g

Result

numP
24

c. Write the unit name and unit number of units located in location C?

Ingres DML:

*range of m is unitmodel

*range of u is unit

*retrieve (m.name, u.num)

*where u.dep = m.dep and u.level = m.level

*and u.type = m.type and u.location = "c"

*\g

Results

Name	Num
Infantry Brigade	200
Infantry Battalion	020
Infantry Battalion	021
Tank Battalion	200

d. How many technicians does the signal corp department supply to the armed forces?

Ingres DML:

```
*range of a is assignp
*range of u is unit
*retrieve into temp (a.dep, a.level, a.type,
   u.num, nump = a.num)
*where a.dep = u.dep and a.level = u.level
*and a.type = u.type, and a.spec = "tec"
*and a.pdep = "16"
*\g
*range of t is temp
*retrieve (tech = sum (t.nump))
*\g
```

Result:

tech
22

d. Write the unit's name and number for the units which are located in location C and in plan C?

Ingres DML:

```
*range of u is unit
*range of m is unitmodel
*range of p is plan
*retrieve into temp (u.dep, u.level, u.type,
 u.num, m.name)
*where u.dep = m.dep and u.level = m.level
```

```
*and u.cype = m.type and u.location = "c"
*\g
*range of t is temp
*retrieve (t.name, t.num)
*where t.dep = p.dep and t.level = P.level
*and t.type = P.type and t.num = P.num
*and P.plan = "c"
*\g
```

Result

Name	Num
Tank battalion	200

The above queries are examples of useful output that may be done by the end user with little training. Let us try to answer other types of questions. For example, how many drivers exist in the formation commanded by the infantry brigade 100?

To answer this query, we first must know what units are existing in this formation. The command relation can give us the set of units that is commanded directly by the infantry brigade 100 (i.e.) one level of the formation hierarchy. Unfortunately, we do not know how many levels we should travel. For that reason the query should have a recursive feature. Unfortunately, neither ingress DML nor equal (embedded ingress in c language) allows recursive constructions. From the above information and to let our

system answer any type of query (about the data in the ten relations), we need an algorithm to collect the units that belong to a particular formation or, in other words, get the nodes of the subtree that belong to a specified node in the armed force tree. This algorithm will be introduced in the next section.

Collect Formation Unit Algorithm.

For the reason of keeping the set of units that is under the command of a particular unit, we create a new relation in the system, the relation formation (dep, level, type, num). This relation is a working relation to keep the unit collection of the required formation (the output of the algorithm). Then we can do our queries about that formation.

The algorithm of collecting the units under the command of a specified unit is:

1. Get the command unit code - unit code is (dep/level/type/number).
2. Create the temporary relation parent (dep, level, type, num).
3. Create the temporary relation sons (dep, level, type, num).
4. Delete the controls of the relation formation (if any).
5. Append to parent the command unit - unit here means unit co

6. Append to sons the units that are commanded directly by any unit(s) which exist in parent (using command relation).
7. Append to formation all units in parent.
8. Delete the contents of parent.
9. If the number of units in sons < 1, go to Step 13 or else continue.
10. Append to parent all units in sons.
11. Delete the contents of sons.
12. Go to Step 6.
13. Destroy sons, destroy parent.
14. Exit.

(See Figure 9)

This algorithm is established as the computer program "tree" (see Appendix C).

To get the set of units that is commanded by a specified unit, from UNIX, type "tree". The program tree will start running and it will let you enter the command unit data (dep, level, type, num) as the program requests. The program will check the existence of the command unit. If the unit does not exist, or if the input format is not correct, the program will ask you to re-enter the command unit data again or to quit (by typing 'Q'). If the entered data are correct, the program will start execution and the output will be stored in the relation formation. If the entered

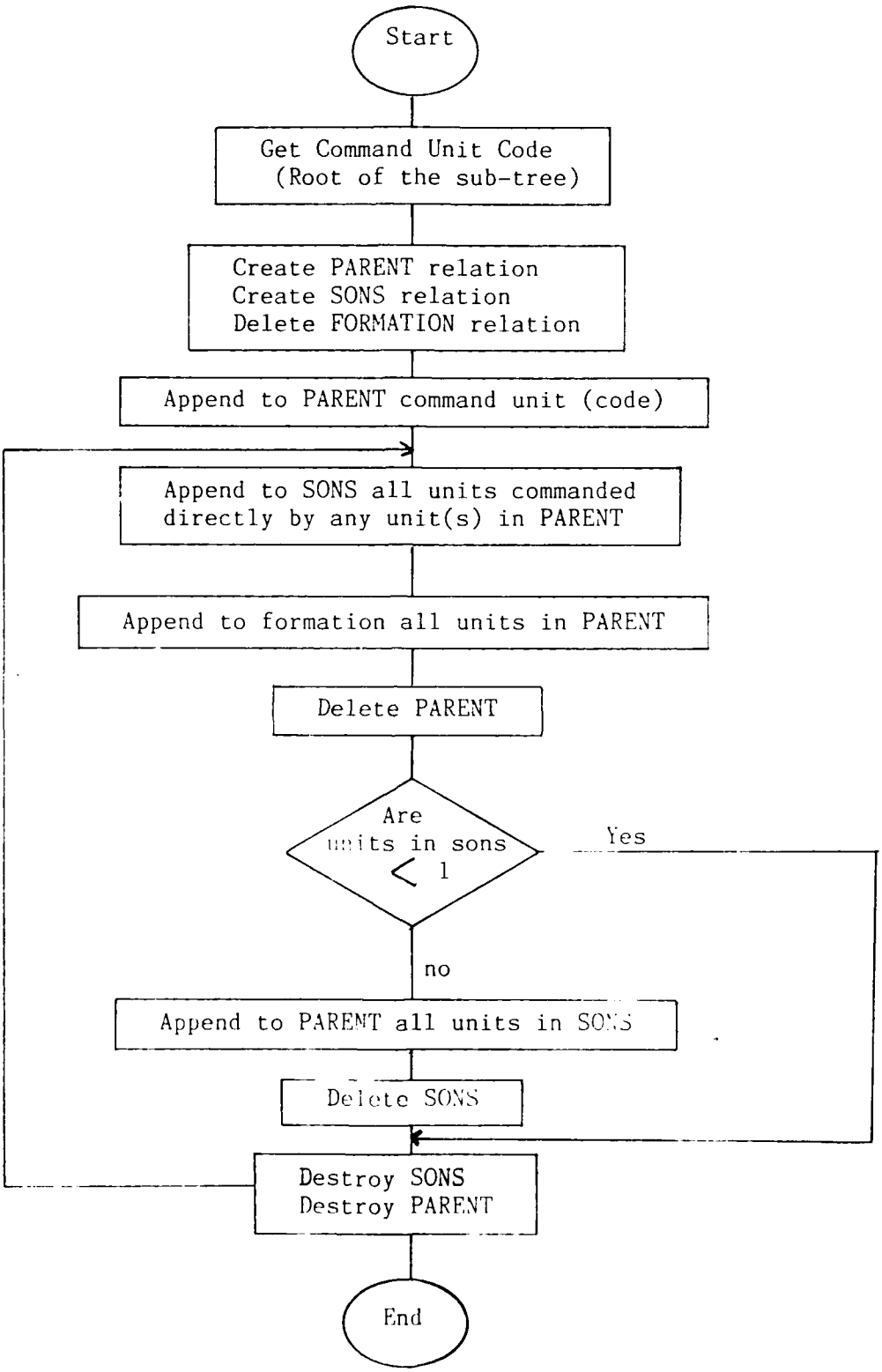


Figure 9. Coll[illegible] Formation Units Algorithm

command unit is a leaf unit (command no unit), the relation formation will contain this unit only and the program will let you know by an appropriate message.

The program (TREE) is done in C language with embedded ingres. It consists of a main body (main) and 3 functions (get-unit-model, check unit, and Ctree) as shown in the structure chart described in Figure 10. TREE program is presented in Appendix C, and a copy of the output results is presented in Appendix D.

The answer to the query described in the previous section (how many drivers exist in the infantry brigade 100) will be presented in Appendix E using tree program.

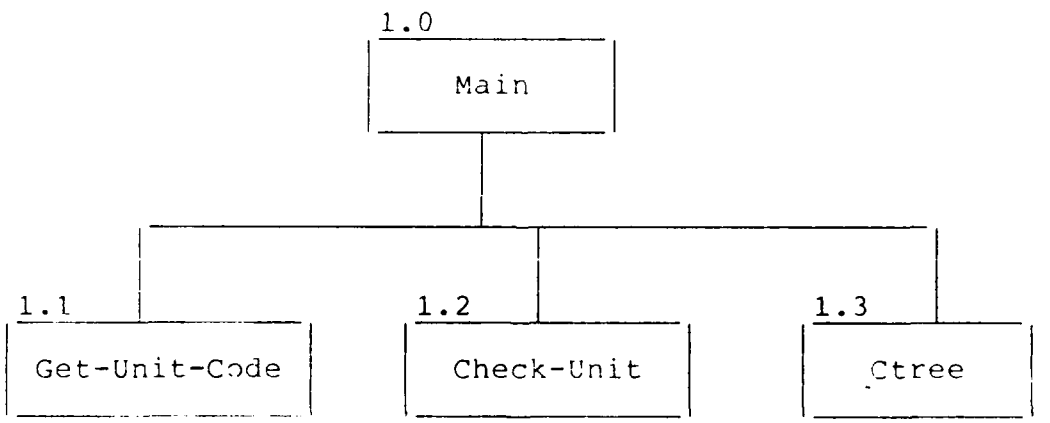


Figure 10. The Structure Chart of Tree Program

Updating of Data

The problem of data updating in our system should be treated carefully. The designed system consists of 10

relations and an update in assigned relation may violate the integrity of the data base. For example, if we are trying to delete a unit, deleting the unit from unit relation is not enough. It should also be deleted from any occurrence in plan relation tuples and also from the occurrence in the command relation as a commanded unit. Another violation of the consistency of data will happen if we delete a unit which exists in the command relation or a command unit. If we only do this, we will leave its subunit be commanded by a non-existing unit. In this case, we should either delete all the units commanded by this unit or quit according to the user request.

The updating of the program "OMAED" which is presented in Appendix G is designed to perform the three main updating functions (insertion, deletion, and modification) in a certain algorithm so that it keeps the data integrity over the whole system. The ability of retrieving unit data is added to the program to let the user retrieve unit data according to his request. The program is designed in menu driven fashion to make it easy to use. The program is designed in main body and 22 functions that count 23 modules. The structure chart of OMAED program is shown in Figure 11.

Insertion.

To insert a new unit into the data base, the program

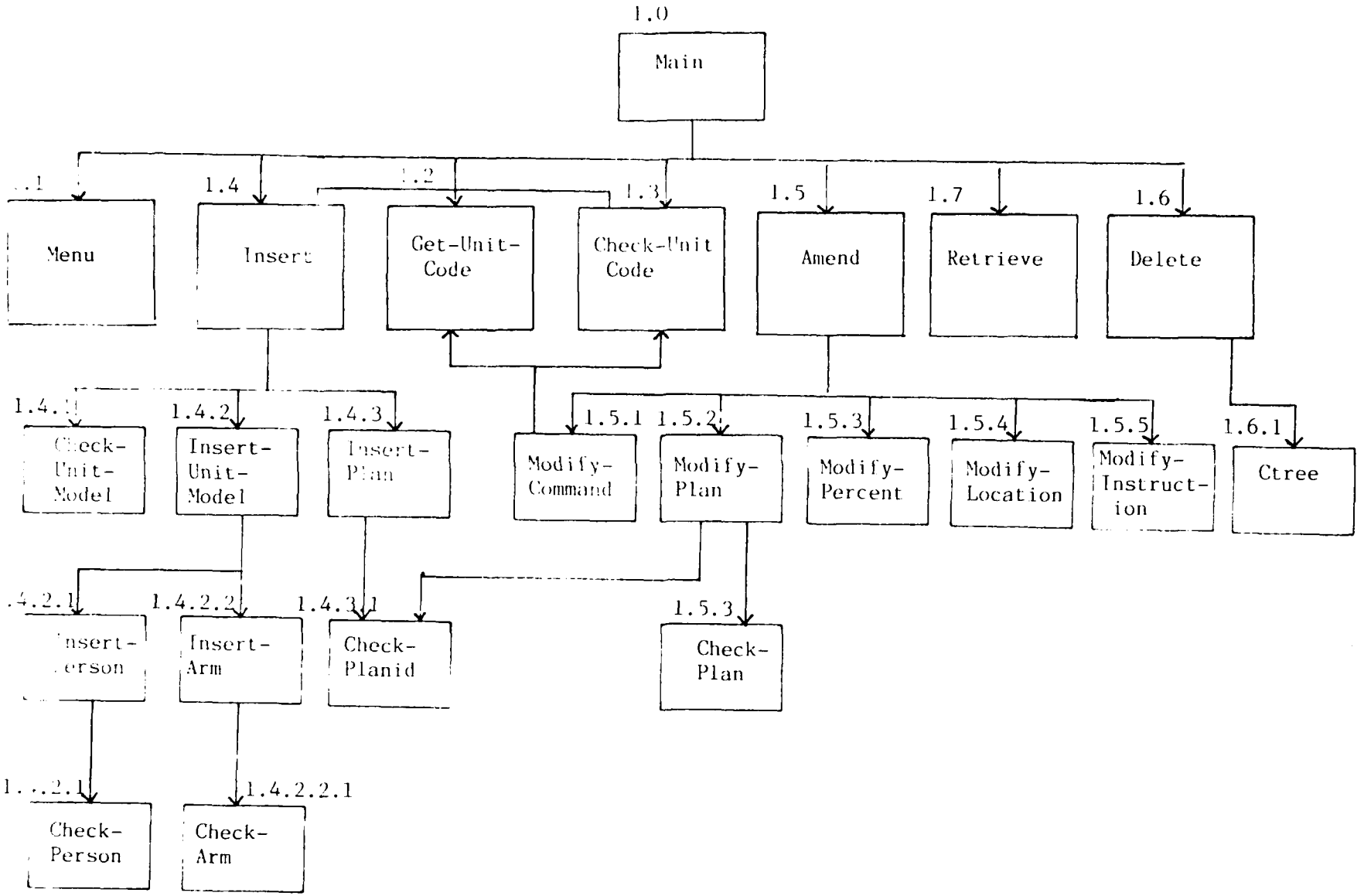


Figure 11. The Structure Chart of OMAED Program

will check the format of the unit code first before calling ingress, then it checks if the unit exists. The system does not allow the existence of the same unit twice. If these checks are passed, the program will check if the new unit has an existing unit model. If not, it will ask the user to enter the new unit model data and it gives the user the chance to quit. After insuring that the unit model exists (or correctly entered by the user), the program will ask the user to enter the unit data (unit number, percentage, location, plan(s), instruction number, and command unit). The system will make an appropriate check for each entered piece of data. For example, the command unit should be an existing unit, plan should exist in planid relation.

After insertion of the data, the program returns to the menu, and the user can retrieve the entered data using "r" option.

Modification.

To modify existing unit data, the program will check first the existence of the unit in the data base, then it will show another menu to select one of the following modification options:

- modify unit percentage
- modify unit location
- modify unit plan
- modify command unit
- modify instruction number

- exit (return to main menu)

For every selection, the program will ask the user to enter the new data, then it makes an appropriate check on it. For example, modification of plan will be done through two operations, either to assign the unit to a new plan or to delete a previous assignment for plan. To do that, there should be two types of checks, one to insure that the new plan exists, the other check is to insure that we will not try to delete a plan which the unit is not assigned to. Modification of command unit will insure that the new command unit exists.

Deletion.

To delete an existing unit, the program will start a check to see if the unit exists. If so, the program will give the user a chance to quit by asking "Are you sure?" At this step, the program cannot delete the unit before doing another operation. The program will be called "Ctree" module to collect the unit(s) which are commanded by the unit in question into the relation "formation". If there is any other unit in the relation formation rather than the unit in question, the program will ask the user again if he wants to delete all subunits. At this step, the user has another chance to quit. If not, the program will delete the unit and all its subunit(s), if any. Deletion is done in such a way as to keep the integrity of data base. That will be discussed in Chapter VIII.

VII. Recovery, Concurrency and Security

Recovery

In our system, the problem of recovery may arise due to the transaction processing. The transaction is a logical unit of work. When executing the data base editing program EDOMADB, one transaction that seems to the user as one operation may be a sequence of several operations. For example, deleting a unit looks like a single transaction, but actually it affects three relations. One tuple must be deleted from unit relation, another tuple must be deleted from command relation, and all the occurrences of this unit in plan relation must be deleted. A similar situation is done when inserting a new unit. A system crash might occur between two operations, or the program itself might, for any reason, terminate between the two. In this case, the program has no means to detect in which stage the failure is done. The recovery should be done on the system level.

A system that supports transaction processing should provide a guarantee that if the transaction executes some updates and then a failure occurs (for whatever reason) before the transaction reaches its normal termination, then those updates will be undone. Thus the transaction either executes in its entirety or is totally canceled (2:414).

The COMMIT and ROLLBACK operations are the key to providing the system recovery. The COMMIT operation indicates to the system that a successful logical unit of work is completed and the data base is in a consistent state. The ROLLBACK operation tells the system that something has gone wrong and a transaction is unsuccessfully terminated. In this case, the data base might be in an inconsistent state, so this logical unit of work should be undone or rolled back. COMMIT and ROLLBACK (and program initiation) represent a synchronization point for the system and at this synchronization point all updates made by the program since the previous synchronization point are committed (COMMIT) or undone (ROLLBACK).

> Note carefully that COMMIT or ROLLBACK terminate the transaction not the program. In general, a single program execution will consist of a sequence of several transactions, running one after the other, with each COMMIT or ROLLBACK operation terminating one transaction and starting the next. (2:416)

Another type of failure is possible. It is the media failure (such as a disk head crash). In this case, the disk has been physically destroyed. Recovery from such failure needs to reload the back-up copy of the data base and uses the log (either manual or in the computer) to redo all transactions that have been done since the previous archive operation for the back-up copy.

Back-up copies should be done frequently and the

data base state reached logged. For example, use the organi- zation instruction number reached or date of last update.

Concurrency

The concurrency problem may arise when the system allows more than one user to update the data base concurrently. In our particular system, concurrency problems may also arise when more than one user is trying to ask questions about the unit formation. In this case, the working relation FORMATION is used to contain the formation units. The sequence of operations is that the program deletes the previous contents of the relation FORMATION, then starts appending to it the unit of the formation under question. After finishing, the required query will be done upon the relation FORMATION. If another user is trying to ask questions about another formation unit, and just after the first user finishes collecting his formation units into the relation FORMATION, second user is trying to delete the relation formation to use it in his query. If he succeeds, the first user will receive false results.

Fortunately, deleting the relation FORMATION is a kind of update and we will return to the problems of concurrent update. There are three concurrency problems:

The Lost Update Problem.

The lost update problem will appear when, for example,

there are two instructions. The first one is to add 3 guns to Unit X, and the second one is to add 4 guns to the same unit. User A is trying to add 3 guns and user B is trying to add 4 guns concurrently. User A reads the number of guns of Unit X in his buffer as 10. Then user B reads the number of guns of Unit X in his buffer as 10. User A adds 3 to 10 and writes the number of guns as 13. User B adds 4 to 10 and writes the number of guns as 14. Finally, Unit X got 14 guns instead of 17.

The Uncommitted Dependency Problem.

The uncommitted dependency problem occurs when a transaction reads updated data from another transaction which is not committed yet and might rollback. For example, in the previous example, if user A adds 3 guns to Unit X but does not commit yet, and User B reads Unit X data as 13 guns, then user A rollsback, then user B adds 4 guns to Unit X and writes the sum as 17 guns instead of 14.

The Inconsistent Analysis Problem.

Suppose Unit Y has 10 officers, 10 secretaries, and 20 technicians. The sum is 40 persons. User A is asking about the sum of personnel in Unit A, he reads 10 officers, and 10 secretaries. Then User B is trying to make an update in Unit Y. He deletes 5 secretaries and adds 5 technicians (the sum should stay at 40). Then User A reads the number of technicians as 25 and adds the sum as 45 instead of 40.

Locking.

A way of solving the concurrency problem is through a locking mechanism. The basic idea of locking is when a transaction needs an access on an item in the data base, it acquires a lock on that object. The effect of the lock is to prevent other transactions from accessing that object before its committment accessing and thus it prevents other transactions from changing it or from getting false data (in case of update then rollback).

There are two types of locks, exclusive locks (x locks) and shared locks (s locks). X lock is requested when the transaction will update the data base. S lock is requested when the transaction will only retrieve the data. Many transactions of s lock may access the same data item. But only one transaction can access the data item if x lock is requested (i.e., x lock can not share any type of lock). A disadvantage of the locking mechanism is the probability of the deadlock. This problem should be handled by the system having the ability to detect the deadlock and decide what to do to recover from it. The whole concurrency problem will be handled on the system level and we should insure that the system is able to solve this problem.

Security

Security is to protect the data base against both undesired modification or destruction of data and against unauthorized reading of data (7:355).

General Considerations.

There are some aspects to the security problem that should be considered:

- Legal and social aspects. For example, if the user asking about the unit data has the right to do so.
- Physical Control. For example, are the computer, the terminal rooms, and the magnetic media locked and/or guarded?
- Policy question. For example, what is the policy of accessing, i.e., who should be allowed to access what?
- Operational problem. For example, how are the passwords that are used kept secret?
- Hardware control. For example, does the CPU provide any security feature, such as storage protection keys?
- Operating system security. For example, does the underlying operating system erase the contents of storage and data when it is finished? (2:238).

View Mechanism.

The view mechanism is an efficient way for security purposes. We can create a view to a user and the user will access the view as if it is a real relation. There are three ways for making a view. First, a user may see the view as a projection of the original relation. For example, the transportation department needs only to know which unit is in which location to manage the transportation plan, so there is no reason to let its user see unit percentage of completion or instruction number. Other relations may not be seen by the user. For example, relations unit model, persons and armament may be hidden from the user of the food department.

The second type of a view allows the user to see a particular selection of the relation. For example, the armor department can see only the units that are supplied by armor personnel. There is no reason to let the armor department user see the organization of the navy units.

The third type is a combination of projection and selection of data in the particular view.

GRANT and REVOKE Mechanism.

> The view mechanism allows the data base to be conceptually divided up into pieces in various ways so that sensitive information can be hidden from unauthorized users. However, it does not allow for the specification of the operations that authorized users may execute against those pieces. (2:441).

GRANT and REVOKE mechanism or the authorization subsystem can do that. The system administrator has full rights to the data base. According to the organization policy, he can GRANT a user rights to access certain pieces of the data base in a certain way. Some users will have the rights to retrieve the data only, others may have the rights to update but not to delete. Other users may have rights for full access to a certain view.

Any authorized user who has certain rights to access the data base can GRANT rights (not exceeding his own rights) to another user. The second user may again GRANT rights (not exceeding his own rights) to a third user. The first user can REVOKE the right he or she granted to the second user, and this operation REVOKEs automatically the rights of the third user. The system should keep track of the authorization subsystem and the system administrator should check that the authorization subsystem is running according to the organization policy.

VIII. Data Base Integrity and System Interfaces

Data Base Integrity

In any designed data base management system, there should be facilities to protect the data base from both incorrect data (for example, the unit percentage of completeness must be between 0 and 100) and inconsistent data (for example when inserting a new unit, the command unit must be an existing unit). These two problems refer to the integrity problem. Integrity means protecting the data base against the misuse by the authorized users.

Integrity Rules.

There are two integrity rules for the relational model as mentioned by C. J. Date (2):

1. Entity Integrity

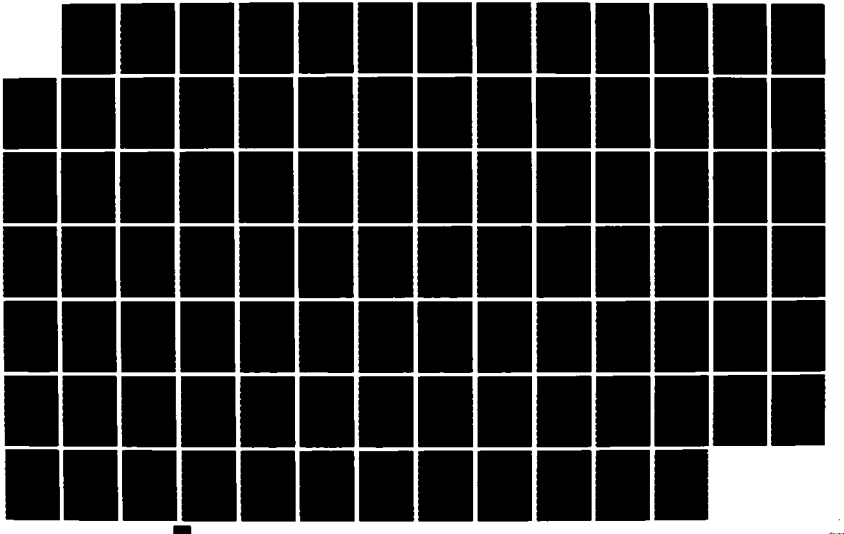
No attribute participating in the primary key of a base relation is allowed to accept null values.

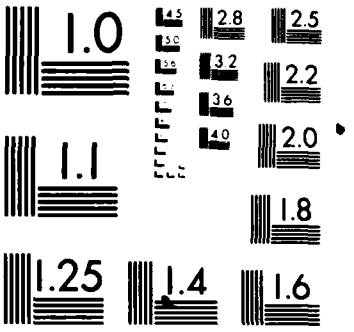
2. Referential Integrity

If base relation R2 includes a foreign key FK matching the primary key PK of some base relation R1, then every value of FK in R2 must either (a) be equal to the value of PK in some tuple of R1 or (b) be wholly null (i.e., each attribute value participating in that FK value must be null). R1 and R2 are not necessarily distinct (2:252).

AD-A172 454 DESIGN AND IMPLEMENTATION OF DATA BASE MANAGEMENT SYSTEM FOR THE ORGANIZA..(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGI.. G A ELSHARAWY 2/2

UNCLASSIFIED DEC 86 AFIT/GCS/ENG/86J-4 F/G 5/9 NL





The second integrity rule is handled in the application program level.

Integrity in OMADB.

Since OMADB consists of 10 relations and or updates in one relation, other relations may be affected. The designed update program EDOMADB observed and handled the second integrity rule carefully.

Inserting a New Unit.

When inserting a new unit, the program starts checking the entered unit code format before calling the data base. If the syntax is correct, the program checks if there is any existing unit in the data base has the entered code. Simply the system cannot allow two units to have the same code. To keep the consistency between UNIT relation and UNITMODEL relation, the program checks if the new unit has an existing unit model. If not, the user has to insert the required unit model before updating UNIT relation. The program asks the user to enter the new unit model data. Three relations (UNITMODEL, ASSIGNP, and ASSIGNA) will be updated to insert the new unit model. To keep the correctness of the unit model itself, the personnel data (person department and specialization) will be checked if such departments provide such specialization or not. The relation that keeps this information is the relation PERSON.

If the department and specialization are consistent, the data will be accepted and a new tuple will be added to the relation ASSIGNP. A similar test is done when entering the unit armament table. The relation armament holds the information about which department supplies which arms or equipment, and the program checks the entered armament data against this relation. If the program insures that the unit model of the new unit exists, it asks the user to enter the unit data to start modifying the unit data which is stored in the relations UNIT, PLAN, and COMMAND. An appropriate test is done on each unit data entered by the user. The unit percentage of completeness must be between 0 and 100. The unit location must be one of the six locations a, b, c, d, e, or f. The unit plan must be identified by PLANID relation and the command unit must be an existing unit. At this point, the system can respond that the new unit is inserted and the data base is in a consistent state.

Amending an Existing Unit Data.

When amending a unit, the program checks the existence of such a unit. The amended module displays a menu to the user for selecting an attribute to be amended. A similar check as is done in case of inserting a new unit data, is done on the amending data. Because the unit-plan relationship is many to many, each tuple in PLAN relation has the information about one plan for one unit. For this

reason, amending unit plan is done through the insert/delete operation. The program prevents the user from adding a plan which the unit has, and from deleting a non existing plan. Modification is done attribute by attribute and the user has the chance to re-amend the same attribute if he or she needs. After the modification of each attribute, the data base will be in a consistent state.

Deletion of Unit(s).

When the user asks the program to delete a unit, the program checks the existence of the unit first, then it starts checking if this unit has another subunit(s). This check is done by calling the CTREE module that uses COMMAND relation as the source of data and FORMATION relation as the destination of the result and other two temporary relations SONS, and PARENT. After execution of CTREE module, FORMATION relation contains the deleted unit and its subunits if any. The program checks if formation relation contains more than one unit and asks the user if he or she wants to delete all subunits or not. The program is not allowed to delete a unit and leaving its subunit(s) commanded by a non-existing unit. If the user answer is "yes", the program deletes any occurrence of the unit(s) that exist in the relation FORMATION from the three relations UNIT, PLAN, and COMMAND. At this point, the data base is in consistent state.

System Interfaces

Recalling Figure 1 which describes the integrated information system for the Egyptian Armed Forces, we will observe that the organization structure system has interfaces to other systems. We will discuss these interfaces as two types of activities, data needed from other systems, and data needed for other systems.

Data Needed From Other System.

The ten relations that our system consists of are not basic system relations. Some of these relations may be a view of a relation created and manipulated by another system. One reason for the integrated information system is to keep the data integrity over all organizations in the armed force and consequently in all information system. The ARMAMENT relation, which could be a view of another relation, which contains more information about armament in the material support system.

ARMAMENT relation (or view) may give us an answer of a question that the organization structure alone is not able to answer. What armaments exist in department X (or in the armed forces) that is not used by any unit. The answer of this query gives us the most recent types of armament that is not yet used by any unit, and also gives us the obsolete type that is out of service and the department did not get rid of yet. A different question may arise, and that is

whether the material support system allows the organization structure system to retrieve this information. If not, how will a new unit that will use a new type of weapons be inserted to the data base without the conflict of the non-existance of the new type of weapons in ARMAMENT relation? Again, the view system should be studied carefully. Similarly, PERSON relation may be a view of another relation in the Personnel System, and PLANID relation may be a view of another relation in Command and Control System. For that reason, editing of the relations PERSON, ARMAMENT, and PLAN is not mentioned in the EDOMADB program. To let the system work, and until establishing the interfaces to other systems, editing of the mentioned three relations will be done by separate programs and we will let that be done by users representing Personnel Material Support, and Command and Control Systems.

Unit percentage of completeness, location, and plan are amended by operational instruction. The unit is created with these three attributes by our system. Amending these attributes should be done through the command and control system (or its representative user for the time being) and it should have the right to amend percentage and location in UNIT relation (only amend, not delete or insert) and should have full access to PLAN relation.

Data Needed for Other Systems.

The organization structure system is able to provide the other systems useful data. For example, the command and control system needs to know information about the command relationship between units; also, other unit information is needed for the command and control system. UNIT, COMMAND, ASSIGNP, and ASSIGNA relations may be seen by command and control system. UNIT, ASSIGNP, COMMAND may be seen by Personnel Systems. UNIT, ASSIGNA, COMMAND may be seen by Material Support System. The Finance System may need information about number of units, total number of personnel for each specification and/or for each department, and may need some information about unit armaments. Interfaces between systems/subsystems should be identified to keep the overall system integrity.

IX. Conclusions and Recommendations

The integrated information system of the Egyptian Armed Force consists of two main systems, Command and Control System and the Logistic Support System. The Logistic Support System consists of four main systems, Organization Structure System, Material Support System, Personnel System, and Finance System. The integrated information system must not be designed and implemented all at the same time. A better way is to design and implement system by system and subsystem by subsystem keeping in mind the design consideration, the system (or subsystem) interfaces, and the overall data base integrity. This projects presents a design model and a methodology that may be followed in other systems and subsystems. One way of doing that is to assign the mission of the system integration to an organization (maybe information systems department). This organization designs the over all system design, determines the design considerations for each system and subsystem, and sets the required specification for each system and subsystem. Once the overall design is done, design and implementation of each system and subsystem should be assigned to an armed force organization taking into consideration the required personnel, training, and finance. To keep the integrated

information system homogeneous, the system integrator should choose the proper DBMS model that will be used in all systems and subsystems. (This project choose the relational data base model for the considerations mentioned in Chapter V). The needed software and hardware should be estimated early by information system departments for the entire integrated information system. The signal corp department may carry the mission of design and implementation of the data communication needed for the integrated system.

Four problems should be handled carefully over all the system and subsystem.

1. Recovery from failure
2. Concurrency
3. System security
4. Data integrity

Again, system/subsystem interfaces should be identified clearly and carefully.

This thesis effort contributes to building the Egyptian Armed Force integrated information system. Design and implementation of other system/subsystem in the integrated information system may follow the same methodology as used in this work.

Appendix A

Ingres Data Definition

```
Script started on Sun May  4 15:48:11 1986
% ingres omada
INGRES version 7.10 (10/27/81) login
Sun May  4 15:49:11 1986
 go
* help
* \g
Executing . . .

 relation name      relation owner

 relation           gsharawy
 attribute          gsharawy
 indexes            gsharawy
 tree               gsharawy
 protect            gsharawy
 integrities        gsharawy
 unitmodel          gsharawy
 department         gsharawy
 person             gsharawy
 armmament          gsharawy
 assignp            gsharawy
 assigna            gsharawy
 unit               gsharawy
 plan               gsharawy
 command            gsharawy
 formation          gsharawy
 planid             gsharawy


continue
* help unitmodel
* \g
Executing . . .

Relation:                unitmodel
Owner:                   gsharawy
Tuple width:             37
Saved until:             Sun Jun 22 01:00:00 1986
Number of tuples:        7
Storage structure:       paged heap
Relation type:           user relation

 attribute name     type  length  keyno.

 deo                 c        2
 level               c        2
 type                c        2
 name                c       30
 balance             c        1


continue
```

```
* help department
* \g
Executing . . .


Relation:                   department
Owner:                      gsharawy
Tuple width:                4
Saved until:                Thu May  1 09:47:33 1986
Number of tuples:           4
Storage structure:          paged heap
Relation type:              user relation


 attribute name     type  length  keyno.

 dep                 c        2
 branch              c        2


continue
* help person
* \g
Executing . . .


Relation:                   person
Owner:                      gsharawy
Tuple width:                5
Saved until:                Sun Jun 22 01:00:00 1986
Number of tuples:           14
Storage structure:          paged heap
Relation type:              user relation


 attribute name     type  length  keyno.

 pdep                c        2
 spec                c        3


continue
* help assignment
* \g
Executing . . .


Relation:                   assignment
Owner:                      gsharawy
Tuple width:                32
Saved until:                Sun Jun 22 01:00:00 1986
Number of tuples:           12
Storage structure:          paged heap
Relation type:              user relation
```

attribute name	type	length	keyno.
adep	c	2	
name	c	30	

```
continue
* help assignp
* \g
Executing . . .
```

```
Relation:                   assignp
Owner:                      gsharawy
Tuple width:                15
Saved until:                Sun Jun 22 01:00:00 1986
Number of tuples:           36
Storage structure:          paged heap
Relation type:              user relation
```

attribute name	type	length	keyno.
dep	c	2	
level	c	2	
type	c	2	
pdep	c	2	
spec	c	3	
num	i	4	

```
continue
* help assigna
* \g
Executing . . .
```

```
Relation:                   assigna
Owner:                      gsharawy
Tuple width:                42
Saved until:                Sun Jun 22 01:00:00 1986
Number of tuples:           22
Storage structure:          paged heap
Relation type:              user relation
```

attribute name	type	length	keyno.
dep	c	2	
level	c	2	
type	c	2	
adep	c	2	
name	c	30	
num	i	4	

```
continue
* help unit
* \g
Executing . . .

Relation:                unit
Owner:                   gsharawy
Tuple width:             19
Saved until:             Sun Jun 22 01:00:00 1986
Number of tuples:        12
Storage structure:       paged heap
Relation type:           user relation

 attribute name     type  length  keyno.

 dep                 c        2
 level               c        2
 type                c        2
 num                 c        3
 percentage          i        4
 location            c        1
 instruction         c        5

continue
* help plan
* \g
Executing . . .

Relation:                plan
Owner:                   gsharawy
Tuple width:             10
Saved until:             Sun Jun 22 01:00:00 1986
Number of tuples:        19
Storage structure:       paged heap
Relation type:           user relation

 attribute name     type  length  keyno.

 dep                 c        2
 level               c        2
 type                c        2
 num                 c        3
 plan                c        1

continue
* help command
* \g
Executing . . .
```

```
Relation:                   command
Owner:                      gsharawy
Tuple width:                18
Saved until:                Sun Jun 22 01:00:00 1986
Number of tuples:           11
Storage structure:          paged heap
Relation type:              user relation


 attribute name     type  length  keyno.

 cdep                c        2
 clevel              c        2
 ctype               c        2
 cnum                c        3
 dep                 c        2
 level               c        2
 type                c        2
 num                 c        3


continue
* help planid
* \g
Executing . . .


Relation:                   planid
Owner:                      gsharawy
Tuple width:                1
Saved until:                Sun Jun 22 01:00:00 1986
Number of tuples:           4
Storage structure:          paged heap
Relation type:              user relation


 attribute name     type  length  keyno.

 id                  c        1


continue
* \q
INGRES version 7.10 (10/27/81) logout
Sun May  4 15:52:07 1986
goodbye gsharawy -- come again
%
Script done on Sun May  4 15:52:10 1986
```

Appendix B

Representation of Sample Data

```
Script started on Sun May  4 15:52:23 1986
% ømgres omadb
INGRES version 7.10 (10/27/81) login
Sun May  4 15:53:24 1986
 go
* print unitmodel
* \g
Executing . . .
```

unitmodel relation

dep	level	type	name	balanc
01	00	00	mod	1
10	01	00	infantry dep	1
11	01	00	armour dep	1
16	01	00	signal corp dep	1
10	05	02	infantry brigade	2
10	06	03	infantry battalion	2
11	06	07	tank battalion	2

```
continue
* print department
* \g
Executing . . .
```

department relation

dep	agency
01	ar
10	ar
11	ar
16	ar

```
continue
* print person
* \g
Executing . . .
```

person relation

pdep	spec
10	off
10	sec
10	com
10	dri
10	cra
11	off
11	sec

```
|11     |com    |
|11     |dri    |
|11     |tec    |
|16     |off    |
|16     |sec    |
|16     |com    |
|16     |tec    |
|-------------|
```

```
continue
* print armmament
* \g
Executing . . .
```

armmament relation

```
|adep   |name                                |
|------------------------------------------|
|10     |pistol 9 mm                         |
|10     |automatic rifle                     |
|10     |machine gun                         |
|10     |jeep 4x4                            |
|10     |truck 4x4 3 tons                    |
|10     |truck 4x4 5 tons                    |
|11     |m113 a2                             |
|11     |ot 62                               |
|11     |tank m60 a3                         |
|11     |tank t62                            |
|16     |wireless set r240                   |
|16     |wireless set r147                   |
|------------------------------------------|
```

```
continue
* print assignp
* \g
Executing . . .
```

assignp relation

```
|dep    |level |type   |pdep   |spec   |num          |
|-----------------------------------------------------|
|01     |00    |00     |10     |off    |           32|
|01     |00    |00     |10     |sec    |           52|
|01     |00    |00     |10     |com    |           30|
|01     |00    |00     |10     |dri    |           16|
|01     |00    |00     |10     |cra    |            6|
|01     |00    |00     |11     |off    |           21|
|01     |00    |00     |16     |off    |           10|
|01     |00    |00     |16     |tec    |           12|
|10     |01    |00     |10     |off    |           16|
|10     |01    |00     |10     |sec    |           20|
|10     |01    |00     |10     |com    |           24|
|10     |01    |00     |10     |dri    |            8|
|10     |01    |00     |16     |tec    |            2|
```

11	01	00	11	off	18
11	01	00	11	sec	21
11	01	00	11	com	20
11	01	00	11	dri	10
11	01	00	16	tec	2
16	01	00	16	off	16
16	01	00	16	sec	12
16	01	00	16	dri	4
16	01	00	16	tec	2
10	05	02	10	off	14
10	05	02	10	com	40
10	05	02	10	dri	5
10	05	02	11	tec	6
10	05	02	16	off	1
10	05	02	16	tec	2
10	06	03	10	off	30
10	06	03	10	com	300
10	06	03	11	dri	20
11	06	07	11	off	24
11	06	07	11	com	72
11	06	07	11	dri	31
11	06	07	11	tec	4

continue
* print assigna
* \g
Executing . . .

assigna relation

dep	level	type	adep	name	num
01	00	00	10	pistol 9 mm	79
01	00	00	10	automatic rifle	100
01	00	00	10	jeep 4x4	6
01	00	00	10	truck 4x4 3 tons	4
10	01	00	10	pistol 9 mm	30
10	01	00	10	automatic rifle	40
10	01	00	10	truck 4x4 3 tons	5
11	01	00	10	pistol 9 mm	30
11	01	00	10	automatic rifle	41
11	01	00	10	truck 4x4 3 tons	5
16	01	00	10	pistol 9 mm	20
16	01	00	10	automatic rifle	14
16	01	00	10	truck 4x4 3 tons	4
10	05	02	10	automatic rifle	330
10	05	02	10	machine gun	20
10	05	02	11	m113 a2	20
10	05	02	10	jeep 4x4	12
10	05	02	10	truck 4x4 5 tons	8
11	06	07	10	pistol 9 mm	131
11	06	07	11	tank m60 a3	31
11	06	07	10	jeep 4x4	4
11	06	07	10	truck 4x4 3 tons	3

```
continue
* print unit
* \g
Executing . . .
```

unit relation

dep	level	type	num	percentage	locati	instru
01	00	00	000	100	a	10001
10	01	00	000	70	a	20010
11	01	00	000	70	a	20011
16	01	00	000	70	a	20016
10	05	02	100	90	b	30100
10	05	02	200	85	c	30200
10	06	03	010	90	b	30100
10	06	03	011	90	b	30100
10	06	03	020	85	c	30200
10	06	03	021	85	c	30600
11	06	07	100	90	b	40100
11	06	07	200	85	c	40100

```
continue
* print plan
* \g
Executing . . .
```

plan relation

dep	level	type	num	plan
01	00	00	000	a
01	00	00	000	b
01	00	00	000	c
01	01	00	000	a
11	01	00	000	a
16	01	00	000	a
10	05	02	100	b
10	05	02	100	c
10	06	03	010	b
10	06	03	010	c
10	06	03	011	b
10	06	03	011	c
11	06	07	100	b
11	06	07	100	c
10	05	02	200	b
10	06	03	020	b
10	06	03	021	b
11	06	07	200	b
11	06	07	200	c

```
continue
* print command
* \g
Executing . . .
```

command relation

cdep	clevel	ctype	cnum	dep	level	type	num
01	00	00	000	10	01	00	000
01	00	00	000	11	01	00	000
01	00	00	000	16	01	00	000
01	00	00	000	10	05	02	100
01	00	00	000	10	05	02	200
10	05	02	100	10	06	03	010
10	05	02	100	10	06	03	011
10	05	02	100	11	06	07	100
10	05	02	200	10	06	03	020
10	05	02	200	10	06	03	021
10	05	02	200	11	06	07	200

```
continue
* print planid
* \g
Executing . . .
```

planid relation

id
a
b
c
d

```
continue
* \q
INGRES version 7.10 (10/27/81) logout
Sun May  4 15:56:16 1986
goodbye gsnarawy -- come again
%
script done on Sun May  4 15:56:19 1986
```

Appendix C

Computer Program for Collecting Formation Units

```
/* ------------------------------------------------------------

            COLLECT FORMATION'S UNITS THROUGH A ROOT
            ---------------------------------------

     LT COL GABER A. ELSHARAWY              AFIT/EN  GCS 86J

------------------------------------------------------------ */
/*************************************************************
DATE     : 7 MAY 1986
VERSION : 1.0
TITLE    : COLLECT UNIT TREE
FILE NAME : TREE.Q
SOFTWARE SYSTEM : UNIX
USE : To collect units that is commanded by a particular
      unit.
CONTENTS : main, get_unit_code, checkunit,ctree.
FUCTION  : 1 - Accept unit code from the terminal
           2 - Check the unit code
           3 - dislay error message if the code is wrong or
               the unit is not exist.
           4 - If the code is correct delete the relation FORMATION
               and using the two temporary relations PARENT and SONS
               append to the relation FORMATION the unit code and the
               all its subunit(s) code(s).

*************************************************************/

/* || 1.0 main */
/*-------------------------------------------------------------
DATE     : 7 MAY 1986
VERSION : 1.0
MODULE NUMBER : 1.0
MODULE NAME   : main
FUNCTION      : Collect Formation's units through a root
ALGORITHM     : 1 - call get_unit_code
                2 - call checkunit
                3 - call ctree function to collect the unit tree
                    into the relation FORMATION
INPUTS        : none
OUTPUTS       : error messages.
GLOBAL VARIABLE READ    : xdep, xlevel, xtype, xnum
GLOBAL VARIABLE CHANGED : none
FILES READ      : none
FILES WRITTEN   : none
HARDWARE INPUT  : none
HARDWARE OUTPUT : none
CALLED  MODULES : get_unit_code(), checkunit(), ctree().
CALLING MODULES : none
AUTHER          : LT COL GABER A. ELSHARAWY
HISTORY         : Version 1.0 by Lt Col Gaber A. Elsharawy
                  AFIT/EN GCS 86J   7 MAY 1986

-------------------------------------------------------------*/

#include <stdio.h>
```

```
## char xdep[3],xlevel[3],xtype[3],xnum[4];
main()
extern char xdep[],xlevel[],xtype[],xnum[];
{
int count,num_units;
  get_unit_code();
   printf(" Calling ingres \n");
## ingres omadb
   printf(" Unit check : ");
     if (checkunit() == 0)
     {
       printf("This unit is not exist .. try again \n");
     }
     else
     {
      printf("Passed\n");
      num_units = ctree();
      if (num_units == 1)
       {
        printf("This relation contains one tuple\n");
       }
      }
## exit
}
/*-----------------------------------------------------------------*/
/* || 1.2 checkunit * /
/*-----------------------------------------------------------------
DATE     : 7 MAY 1986
VERSION  : 1.0
MODULE NUMBER : 1.2
MODULE NAME   : checkunit()
FUNCTION      : check the unit code(dep,level,type,num).
ALGORITHM     : 1 - if the unit code exist in the relation UNIT
                    return(1)
                2 - if not return(0).
INPUTS        : none
OUTPUTS       : 0 or 1
GLOBAL VARIABLE READ    : xdep, xlevel, xtype, xnum
GLOBAL VARIABLE CHANGED : none
FILES READ    : none
FILES WRITTEN : none
HARDWARE INPUT : none
HARDWARE OUTPUT: none
CALLED  MODULES : main()
CALLING MODULES : none
AUTHER          : LT COL GABER A. ELSHARAWY
HISTORY         : Version 1.0 by Lt Col Gaber A. Elsharawy
                  AFIT/EN GCS 86J   7 MAY 1986

-----------------------------------------------------------------*/

checkunit()
{
/* extern char xdep[],xlevel[],xtype[],x . . ]*/
## char tdep[3], tlevel[3], ttype[3], t . . .
## range of u is unit
```

```
** retrieve(tdep = u.dep, tlevel = u.level, ttype = u.type,
**           tnum = u.num)
**           where  u.dep   = xdep
**           and    u.level = xlevel
**           and    u.type  = xtype
**           and    u.num   = xnum
  if (strcmp(xdep, tdep) == 0
  &&  strcmp(xlevel, tlevel) == 0
  &&  strcmp(xtype, ttype) == 0
  &&  strcmp(xnum, tnum) == 0)
    {
     return(1);
    }
  else
    {
    return(0);
    }
}
/*-------------------------------------------------------------------*/
/* || 1.3 main * /
/*-------------------------------------------------------------------
DATE     : 7 MAY 1986
VERSION : 1.0
MODULE NUMBER : 1.3
MODULE NAME   : ctree()
FUNCTION      : Collect formation's unit through a root.
ALGORITHM     : 1 - Create the temporary relation PARENT
                2 - Create the temporary relation SONS
                3 - Delete the contents of the relation FORMATION
                4 - Append to PARENT the command unit (from
                    the global variables, xdep,xlevel,xtype,xnum)
                5 - Append to SONS unit that is commanded directly
                    by any unit in PARENT(using the relation COMMAND)
                6 - Append to FORMATION all units in PARENT.
                7 - Delete the contents of PARENT.
                8 - If number of units in SONS < 1 then EXIT
                    else continue.
                9 - Append to Parent all units in SONS.
                10- Delete the contents of SONS
                11- Go to step 5
                12- Destroy SONS, destroy PARENT, exit.
INPUTS        : Root unit code
OUTPUTS       : The ralation FORMATION
GLOBAL VARIABLE READ    : xdep, xlevel, xtype, xnum
GLOBAL VARIABLE CHANGED : none
FILES READ    : none
FILES WRITTEN : none
HARDWARE INPUT : none
HARDWARE OUTPUT: none
CALLED  MODULES :
CALLING MODULES : none
AUTHER          : LT COL GABER A. ELSHA-
HISTORY         : Version 1.0 by Lt Col Gaber A. Elsharaby
                  AFIT/EN GCS 86J   7 MAY 1986

----------  ----------  ----------  ----------*/
```

```
ctree()
{
/* **extern char xdep,xlevel,xtype,xnum */
**int  flag,f1,f2,f3,f4;
** create parent(dep= c2,level= c2,type= c2,num= c3)
** create  sons(dep= c2,level= c2,type= c2,num= c3)
** range of p is parent
** range of s is sons
** range of f is formation
** range of c is command
** delete f
** append to parent(dep = xdep,level = xlevel,type = xtype,
**                        num = xnum)
do {
** append to sons(dep = c.dep,level= c.level,type= c.type,
**                       num= c.num)
**             where  c.cdep = p.dep
**               and  c.clevel = p.level
**               and  c.ctype  = p.type
**               and  c.cnum   = p.num
** append to formation (p.all)
** delete p
** retrieve (flag = count(s.dep))
   if(flag < 1)
   {
   printf("Relation FORMATION(dep,level,type,num) is created \n");
   }
  else
     {
**   append to parent (s.all)
**   delete s
     }
   }
 while (flag > 0);
** retrieve (f1 = count(f.dep))
** retrieve (f2 = count(f.level))
** retrieve (f3 = count(f.type))
** retrieve (f4 = count(f.num))
** destroy parent
** destroy sons
   if(f1 > 1 || f2 > 1 || f3 > 1 || f4 > 1)
      return(2);
   else
     return(1);
}
/*-------------------------------------------------------------------*/
/* || 1.1 get_unit_code * /
/*-------------------------------------------------------------------
DATE    : 7 MAY 1986
VERSION : 1.0
MODULE NUMBER : 1.0
MODULE NAME   : get_unit_code()
FUNCTION      : Accept unit code from user and check its syntax.
ALGORITHM     : 1 - Display messages to let the user enter
                    the unit data(xdep,xlevel,xtype,xnum)
```

```
                    2 - Accept the user input
                    3 - Check the syntax of the usre input
                    4 - If correct exit
                    5 - If not request the user to reinput the data
                        or quit.
INPUTS          : user input(xdep,xlevel,xtype,xnum)
OUTPUTS         : Messeges
GLOBAL VARIABLE READ    : none
GLOBAL VARIABLE CHANGED : xdep, xlevel, xtype, xnum.
FILES READ      : none
FILES WRITTEN   : none
HARDWARE INPUT  : none
HARDWARE OUTPUT : none
CALLED  MODULES : main()
CALLING MODULES : none
AUTHER          : LT COL GABER A. ELSHARAWY
HISTORY         : Version 1.0 by Lt Col Gaber A. Elsharawy
                  AFIT/EN GCS 86J   7 MAY 1986

-------------------------------------------------------------------*/

get_unit_code()
{
/*extern char xdep[],xlevel[],xtype[],xnum[]*/
int reenter;
do
{
  printf("department => ");
  scanf("%s",xdep);
  printf("\n");
  printf("level ======> ");
  scanf("%s",xlevel);
  printf("\n");
  printf("type =======> ");
  scanf("%s",xtype);
  printf("\n");
  printf("number =====> ");
  scanf("%s",xnum);
  printf("\n");
  if ((strlen(xdep) != 2) || (strlen(xlevel) != 2) ||
      (strlen(xtype) != 2) || (strlen(xnum) != 3))
    {
      reenter = 0;
      printf("The key formats should be c2, c2, c2, c3 \n");
      printf("To exit print 2 \n");
          if (xdep[0] == '2')
          {
          exit(1);
          }
      }
  else
    {
    reenter = 1;
    }
\
while (reenter != 1);
```

```
 return;
}
/*-------------------------------------------------------------
                THE END OF THE PROGRAM TREE

   WRITTEN BY LT. COL GABER A. ELSHARAWY    AFIT/EN GCS 86J
------------------------------------------------------------- */
```

Appendix D

Test Results of Collecting Formation Units Program

```
Script started on Mon May  5 22:06:33 1986
% tree
department => 999

level ======> 10

type =======> 110

number =====> 11

The key formate should be c2, c2, c2, c3
To exit print 2
department => 99

level ======> 88

type =======> 77

number =====> 666

 Calling ingres
 Unit check : there is no such nuit .. try again
% tree
department => 10

level ======> 05

type =======> 02

number =====> 100

 Calling ingres
 Unit check : Passed
Relation FORMATION(dep,level,type,num) is created
% ingres$ omadb
INGRES version 7.10 (10/27/81) login
Mon May  5 22:13:57 1986
 go
* print formation
* \g
Executing . . .


formation relation

|dep    |level |type  |num    |
|-----------------------------|
|10     |05    |02    |100    |
|10     |06    |03    |010    |
|10     |06    |03    |011    |
|11     |06    |07    |100    |
|-----------------------------|

continue
* \q
INGRES version 7.10 (10/27/81)  logout
```

```
Mon May  5 22:14:59 1986
goodbye gsharawy -- come again
% inee
department => 11

level ======> 06

type =======> 07

number =====> 100

 Calling ingres
 Unit check : Passed
Relation FORMATION(dep,level,type,num) is created
This relation contains one tuple
% ingres omadb
INGRES version 7.10 (10/27/81) login
Mon May  5 22:19:10 1986
 go
* print formation
* \g
Executing . . .


formation relation

|dep    |level |type  |num    |
|-----------------------------|
|11     |06    |07    |100    |
|-----------------------------|

continue
* \q
INGRES version 7.10 (10/27/81) logout
Mon May  5 22:20:03 1986
goodbye gsharawy -- come again
%
script done on Mon May  5 22:20:08 1986
```

Appendix E

The answer to the query

"How many drivers exist in infantry brigade 100?"

```
Script started on Tue May  6 20:20:53 1986
% tree
department => 10

level ======> 05

type =======> 02

number =====> 100

 Calling ingres
 Unit check : Passed
Relation FORMATION(dep,level,type,num) is created
% ingres omadb
INGRES version 7.10 (10/27/81) login
Tue May  6 20:25:52 1986
 *
* print formation
* \g
Executing . . .
```

formation relation

dep	level	type	num
10	05	02	100
10	06	03	010
10	06	03	011
11	06	07	100

```
continue
* print assignp
* \g
Executing . . .
```

assignp relation

dep	level	type	pdep	spec	num
01	00	00	10	off	32
01	00	00	10	sec	52
01	00	00	10	com	30
01	00	00	10	dri	16
01	00	00	10	cra	6
01	00	00	11	off	21
01	00	00	16	off	10
01	00	00	16	tec	12
10	01	00	10	off	16
10	01	00	10	sec	20
10	01	00	10	com	14
10	01	00	10	dri	8
10	01	00	16	tec	2
11	01	00	11	off	18

```
|11     |01     |00     |11     |sec    |              21|
|11     |01     |00     |11     |com    |              20|
|11     |01     |00     |11     |dri    |              10|
|11     |01     |00     |16     |tec    |               2|
|16     |01     |00     |16     |off    |              16|
|16     |01     |00     |16     |sec    |              12|
|16     |01     |00     |16     |dri    |               4|
|16     |01     |00     |16     |tec    |               2|
|10     |05     |02     |10     |off    |              14|
|10     |05     |02     |10     |com    |              40|
|10     |05     |02     |10     |dri    |               5|
|10     |05     |02     |11     |tec    |               6|
|10     |05     |02     |16     |off    |               1|
|10     |05     |02     |16     |tec    |               2|
|10     |06     |03     |10     |off    |              30|
|10     |06     |03     |10     |com    |             300|
|10     |06     |03     |11     |dri    |              20|
|11     |06     |07     |11     |off    |              24|
|11     |06     |07     |11     |com    |              72|
|11     |06     |07     |11     |dri    |              31|
|11     |06     |07     |11     |tec    |               4|
|-------------------------------------------------------|

continue
* range of f is formation
* range of p is assgno
* retrieve(drivers = sum(p.num
* where p.dep = f.dep and p.level = f.level
*   and p.type=f.type and p.spec  = "dri"))
* \g
Executing . . .


|drivers       |
|--------------|
|            76|
|--------------|
(1 tuple)

continue
* \q
INGRES version 7.10 (10/27/81) logout
Tue May  6 20:30:15 1986
goodbye gshan .. -- come again
%
script done on Tue May  6 20:30:19 1986
```

Appendix F

User Manual for the Edit Program

Introduction

This manual contains a brief introduction on how to use the edit program OMAED to edit the data base OMADB. It does not go into a lot of detail about the program design or data base features, but tries to give enough information to get a user to use the system. After reviewing this manual, the user should have enough information to deal with the editing program OMAED. The manual is divided into four sections. Each section describes an operation the program can perform (Retrieve, Delete, Amend, and Insert). To get the program started, just type OMAED and hit return, the program will start by displaying "Calling Ingres", then the following menu will be displayed with a command summary

Amending Units Data.

Command Summary.

I . . . . . Insert new unit.

D . . . . . Delete an existing unit.

A . . . . . Amend an existing unit.

R . . . . . Retrieve an existing unit.

H . . . . . Help (display command summary).

E . . . . . Exit to UNIX.

The program can accept the command in lower or upper

case characters. "H" command displays the command summary menu. Now you can select the required operations by entering the corresponding command. After finishing the edit, type "e" to exit to UNIX.

Retrieve Unit Data

To retrieve unit data, type "R" and hit return. The program will ask you to enter unit key (dep, level, type, num) and will check the syntax. If not correct, you will see the message "the key format should be C2, C2, C2, C3." If the unit does not exist in the data base, you will see the message "This unit does not exist." If the key is correct, the system will print on the screen the required unit data (unit name, unit location, instruction number, unit balance, percentage, number of personnel, command unit code, and unit plan(s)). The program will return after that to the command mode.

Deleting a Unit(s) From the Data Base

To delete a unit type "D", the program will ask you to enter the unit key. If the unit does not exist, the program will print "This unit does not exist." The program will ask you "Are you sure (Y/N)?" to give you a chance to quit. If you type any character rather than "Y", the program will return to the command mode. If you type "Y", the

program will delete the unit. If the unit has another subunit, the program will ask you "Do you want to delete all subunits (Y/N)." If you type "Y" the program will delete the unit and all its subunits.

Amending Unit Data

To amend unit data type "A" from the command mode, the program will display the following menu:

Modification Codes.

Modify unit percentage . . . t

Modify unit location . . . l

Modify unit plan . . . . . p

Modify command unit c

Modify instruction number. . i

End of Modification . . . . e

Again the program will accept both lower and upper case letters as modification codes. Select the required modification code and type it. The program will ask you to enter the new unit attribute. After entering the new value, the program will check it to insure that it is in the permissible value domain, if it is not correct, you will receive an error message. After each amend, the program will ask you to enter another modification code, when you finish modification type "e" or "E" to return to the command mode.

Inserting a New Unit

To insert a new unit type "i" or"I", the system will ask you to enter the new unit code. If this unit already exists, you will see the message "This unit already exists." If the new unit has an existing unit model in the data base, the program will ask you to enter the unit data (percentage, location, instruction number, plan(s), and command unit).

Checks will be done on unit data and an error message is displayed if the check is not passed. If the unit has no unit model in the data base, the program will ask you "Do you want to insert a new unit model (Y/N)". This is a chance to quit if you mistyped the unit code. If you type "Y" the program will ask you to enter the following unit model data:

- Unit name
- Personnel data (
  (to end entering
- Armament data (
  (to end entering armament data type "*").

Checks will be done on unit model data and an error message is displayed if the check is not passed. After entering the unit model data, the program will ask you to enter the unit data as described above, then the program will return to the

Appendix G

Computer Program for Editing the Data Base

```
/* ------------------------------------------------------------

            UPDATING OF THE EGYPTIAN ARMED FORCES
            -------------------------------------
              ORGANIZATION STRUCTURE DATA BASE
              --------------------------------
     LT COL GABER A. ELSHARAWY              AFIT/EN  GCS 86J

------------------------------------------------------------ */
/*************************************************************
DATE     : 7 MAY 1986
VERSION : 1.0
TITLE    : EDIT OMADB
FILE NAME : OMAED.Q
SOFTWARE SYSTEM : UNIX
USE : To edit the DBMS OMADB
CONTENTS : main, menue, get_unit_code, checkunit, insert,
           check_unit_model, insert_unit_model, insert_person,
           check_person, insert_arm, check_arm, insert_plan,
           check_planid, ammend, modify_command, modify_plan,
           check_plan, modify_percent, modify_location,
           modify_inst, retrieve, delete, ctree.
FUCTION  :
           1 - Display command summery
           2 - Accept input command(I,D,A,R,M,or E)
           3 - Accept unit code from the terminal
           4 - Check the unit code
           5 _ Call a suitable module
           6 - display error message if the code is wrong or
               if the operation is inconsistanse with the unit
               status(exist/not exist)
           7 - After performing the operation, return to command
               mode.

*************************************************************/

/* 1.0 main */
/*----------------------------------------------------------
DATE     : 7 MAY 1986
VERSION : 1.0
MODULE NUMBER : 1.0
MODULE NAME   : main
FUNCTION      : call ingres, display command summery,
                accept the user command, check it,and
                call a suitable module.
ALGORITHM     : 1 - declear external variables
                2 - call ingres
                3 - display command summery
                4 - accept the user command
                5 - if the command is not correct
                    display an error message and
                    go to step 4
                6 - if the command is d,D,r,R,i,I,
                    a, or A, call get_unit_code
                    and then call check_unit_code.
                7 - unit status = check_unit_code(true/false)
```

```
                    8 - if the unit status is consistence with
                        the operatin, call the required module
                    9 - if not print an error messege
                    10- go to step 4.
INPUTS              : user command
OUTPUTS             : messeges
GLOBAL VARIABLE READ    : none
GLOBAL VARIABLE CHANGED : none
FILES READ          : none
FILES WRITTEN       : none
HARDWARE INPUT      : none
HARDWARE OUTPUT     : none
CALLED  MODULES     :
CALLING MODULES     : none
AUTHER              : LT COL GABER A. ELSHARAWY
HISTORY             : Version 1.0 by Lt Col Gaber A. Elsharawy
                      AFIT/EN GCS 86J   7 MAY 1986

-------------------------------------------------------------*/

#include <stdio.h>
#define true 1
#define false 0
## char xdep[3],xlevel[3],xtype[3],xnum[4];
## char ydep[3],ylevel[3],ytype[3],ynum[4];
## char xplan[2], xpdep[3], xsec[4],xadep[3], xname[31];
## char request[2];
## int state, xnump, xnuma;
/*************************************************************/
main()
{
extern char xdep[],xlevel[],xtype[],xnum[];
extern char ydep[],ylevel[],ytype[],ynum[];
extern char xplan[],xpdep[],xsec[],xadep[],xname[];
extern char request[];
extern int state, xnump, xnuma;
int quit;
printf("Calling ingres\n");
## ingres omadb
   request[0] = 'n';
   request[1] = '\0';
   quit = false;
do
  {
       switch(request[0])
       {
         case 'i' :
         case 'I' : if(state == true)
                       printf("This unit is already exist\n");
                    else
                       insert();
                    break;
         case 'd' :
         case 'D' : if (state == false)
                       printf("This unit is not exist\n");
                    else
```

```
                        delete();
                    break;
            case 'a' :
            case 'A' : if(state == false)
                        printf("This unit is not exist\n");
                    else
                        ammend();
                    break;
            case 'r' :
            case 'R' : if(state == false)
                        printf("This unit is not exist\n");
                    else
                        retrieve();
                    break;
            case 'h' :
            case 'H' : menue();
                    break;
            case 'e' :
            case 'E' : quit = true;
                    break;
            case 'f' : break;
            default  : printf("unrecognized input .. type h for help\n");
                    break;
          }
          if (request[0] != 'E' && request[0] != 'e')
           {
             printf("ENTER COMMAND ==> ");
             scanf("%s",request);
             printf("\n\n");
             if(request[0]=='I' ||request[0]=='i' ||
                request[0]=='D' ||request[0]=='d' ||
                request[0]=='A' ||request[0]=='a' ||
                request[0]=='R' ||request[0]=='r')
                {
                  printf("Enter unit code : \n\n");
                  get_unit_code();
                  state = check_unit_code();
                }
           }
      }
       while (quit != true);
       printf(" GOODBAY COME AGAIN \n");
    exit
}

/* || 1.1 menue */
/*-------------------------------------------------------------------
DATE     : 7 MAY 1986
VERSION  : 1.0
MODULE NUMBER : 1.1
MODULE NAME   : menue
FUNCTION      : Display command summery
ALGORITHM     : Display command summery
INPUTS        : none
OUTPUTS       : Print command summery
GLOBAL VARIABLE READ    : none
```

```
GLOBAL VARIABLE CHANGED : none
FILES READ     : none
FILES WRITTEN : none
HARDWARE INPUT : none
HARDWARE OUTPUT: none
CALLED  MODULES : main
CALLING MODULES : none
AUTHER          : LT COL GABER A. ELSHARAWY
HISTORY         : Version 1.0 by Lt Col Gaber A. Elsharawy
                  AFIT/EN GCS 86J   7 MAY 1986

-----------------------------------------------------------*/

mnue()
{
printf("\n\n\n");
printf("                      AMMENDING UNITS DATA     \n");
printf("                      ---------------------    \n\n");
printf("  COMMAND SUMMERY : \n");
printf("  ----------------- \n\n");
printf("         I ....... Insert new unit. \n");
printf("         D ....... Delete an existing unit. \n");
printf("         A ....... Ammend an existing unit data. \n");
printf("         R ....... Retrieve an existing unit data. \n");
printf("         H ....... Help(display command summery). \n");
printf("         E ....... Exit to UNIX. \n\n");
return;
}

/* || 1.3 check_unit_code * /
/*-------------------------------------------------------------------
DATE     : 7 MAY 1986
VERSION : 1.0
MODULE NUMBER : 1.3
MODULE NAME    : check_unit_code
FUNCTION       : Check the unit code
ALGORITHM      : 1 - if the unit code exist in relation UNIT
                     return(1)
                 2 - if not return(0)
INPUTS         : none
OUTPUTS        : 0 or 1
GLOBAL VARIABLE READ     : xdep, xlevel, xtype, xnum
GLOBAL VARIABLE CHANGED : none
FILES READ     : none
FILES WRITTEN : none
HARDWARE INPUT : none
HARDWARE OUTPUT: none
CALLED  MODULES : main, insert,and modify_command.
CALLING MODULES : none
AUTHER          : LT COL GABER A. ELSHARAWY
HISTORY         : Version 1.0 by Lt Col Gaber A. Elsharawy
                  AFIT/EN GCS 86J   7 MAY 1986

-----------------------------------------------------------*/
check_unit_code()
{
```

```
/* extern char xdep[],xlevel[],xtype[],xnum[]*/
** char tdep[3], tlevel[3], ttype[3], tnum[4];
** range of u is unit
** retrieve(tdep = u.dep, tlevel = u.level, ttype = u.type,
**          tnum = u.num)
**          where  u.dep   = xdep
**          and    u.level = xlevel
**          and    u.type  = xtype
**          and    u.num   = xnum
  if (strcmp(xdep, tdep) == 0
  &&  strcmp(xlevel, tlevel) == 0
  &&  strcmp(xtype, ttype) == 0
  &&  strcmp(xnum, tnum) == 0)
    {
     return(1);
    }
  else
    {
    return(0);
    }
}

/* || 1.2 get_unit_code * /
/*-------------------------------------------------------------------
DATE    : 7 MAY 1986
VERSION : 1.0
MODULE NUMBER : 1.2
MODULE NAME   : get_unit_code
FUNCTION      : 1 - accept unit code from the terminal
                2 - check its syntax
ALGORITHM     : 1 - Display messege to request unit code
                    (dep,level,type,num).
                2 - Accept the unes input
                3 - Check unit code syntax
                4 - If correct exit
                5 - If not request the user to reinput
                    the unit code or quit.
INPUTS        : xdep,xlevel,xtype,xnum
OUTPUTS       : messages
GLOBAL VARIABLE READ    :
GLOBAL VARIABLE CHANGED : xdep, xlevel, xtype, xnum.
FILES READ    : none
FILES WRITTEN : none
HARDWARE INPUT  : none
HARDWARE OUTPUT : none
CALLED  MODULES : main, insert, modify_command.
CALLING MODULES : none
AUTHER          : LT COL GABER A. ELSHARAWY
HISTORY         : Version 1.0 by Lt Col Gaber A. Elsharawy
                  AFIT/EN GCS 86J   7 MAY 1986

-------------------------------------------------------------------*/
get_unit_code()
{
/*extern char xdep[],xlevel[],xtype[],xnum[]*/
int number;
```

```
reenter = 1;
do
{
  printf("department => ");
  scanf("%s",xdep);
  printf("\n");
  if (xdep[0] == 'q' || xdep[0] == 'Q')
      exit(1);
  printf("level ======> ");
  scanf("%s",xlevel);
  printf("\n");
  printf("type =======> ");
  scanf("%s",xtype);
  printf("\n");
  printf("number =====> ");
  scanf("%s",xnum);
  printf("\n");
  if ((strlen(xdep) != 2) || (strlen(xlevel) != 2) ||
      (strlen(xtype) != 2) || (strlen(xnum) != 3))
    {
      reenter = 0;
      printf("The key formate should be c2, c2, c3 \n");
      printf("To exit print Q \n");
    }
  else
    {
    reenter = 1;
    }
}
 while (reenter != 1);
 return;
}

/* || 1.4 insert * /
/*-------------------------------------------------------------------
DATE     : 7 MAY 1986
VERSION  : 1.0
MODULE NUMBER : 1.4
MODULE NAME   : insert
FUNCTION      : Insert new unit into CMAD3
ALGORITHM     : 1 - check the unit code (xdep,xlevel,xtype,xnum)
                2 - if the unit code is not exist in UNITMODEL relation
                    by calling check_unit_model aske the user if he
                    or she wants to insrt a new unit model.
                3 - according to the user request call insert_unit_model
                    or return.
                4 - accept unit data from the terminal (percentage,
                    location, instruction number)
                5 - check the unit data, and if any not in its domain
                    aske the user to reenter it.
                6 - call insert_plan
                7 - ammend the relations UNIT, COMMAND
                8 - return.
INPUTS        : unit data
OUTPUTS       : messages.
GLOBAL VARIABLE READ    : xdep, xlevel, xtype, xnum.
```

```
                                ydep, yleval, ytype, unum.
GLOBAL VARIABLE CHANGED : none
FILES READ      : none
FILES WRITTEN : none
HARDWARE INPUT : none
HARDWARE OUTPUT: none
CALLED  MODULES : main
CALLING MODULES : check_unit_model, insert_unit_model, insert_plan
AUTHER          : LT COL GABER A. ELSHARAWY
HISTORY         : Version 1.0 by Lt Col Gaber A. Elsharawy
                  AFIT/EN GCS 86J   7 MAY 1986

-------------------------------------------------------------------*/
insert()
{
** int reenter, upercentage;
** char ulocation[2], uinstruction[6],res[2];
   if (check_unit_model() == false)
     {
     printf("Do you want to insert a new unit model (y/n) :");
     scanf("%s",res);
     printf("\n");
     if(res[0] != 'y')
       return;
     else
      insert_unit_model();
     }
   printf("Enter unit data : \n\n");
   do
    {
     printf("Percentage : ");
     scanf("%3d",&upercentage);
     printf("\n");
     if (upercentage > 100)
         printf("percentage must be <= 100\n");
    }
   while (upercentage > 100);
   do
    {
     printf("Location : ");
     scanf("%s", ulocation);
     printf("\n");
     if ((strlen(ulocation) == 1) &&
         ulocation[0] == 'a' || ulocation[0] == 'b' ||
         ulocation[0] == 'c' || ulocation[0] == 'd' ||
         ulocation[0] == 'e' || ulocation[0] == 'f')
       reenter = false;
     else
       {
       printf("Unrecognized location \n");
       reenter = true;
       }
    }
   while (reenter == true);
   printf("Instruction number : ");
   scanf("%s",&uinstruction);
```

```
   printf("\n");
** append to unit (dep = xdep, level = xlevel, type = xtype,
**                  num = xnum, percentage = upercentage,
**                  location = ulocation, instruction = uinstruction)
/* The unit relation is updated now */
   insert_plan();
/* The plan relation is updated now */
/* Save unit code to use get_unit_code module to enter
   the command unit code */
   strcpy(ydep,xdep);
   strcpy(ylevel,xlevel);
   strcpy(ytype,xtype);
   strcpy(ynum,xnum);
   do
    {
     printf("Inter command unit data : \n\n");
     get_unit_code();
     if(check_unit_code() == false)
      {
       printf("This unit is not exist\n");
       reenter = true;
      }
     else
       reenter = false;
    }
   while (reenter == true);
** append to command (cdep=xdep, clevel=xlevel, ctype=xtype,
**   cnum=xnum, dep=ydep, level=ylevel, type=ytype, num=ynum)
   return;
  }

/* || 1.4.1 check_unit_model */
/*-------------------------------------------------------------
DATE     : 7 MAY 1986
VERSION  : 1.0
MODULE NUMBER : 1.4
MODULE NAME   : check_unit_model
FUNCTION      : check if the unit model code (xdep,xlevel,xtype)
                is exist in the data base.
ALGORITHM     : 1 - if the unit model code exist in the relation
                    UNITMODEL return(1)
                2 - if not return(0)
INPUTS        : none
OUTPUTS       : 1 or 0
GLOBAL VARIABLE READ    : xdep, xlevel, xtype.
GLOBAL VARIABLE CHANGED : none
FILES READ    : none
FILES WRITTEN : none
HARDWARE INPUT : none
HARDWARE OUTPUT: none
CALLED  MODULES : insert
CALLING MODULES : none
AUTHOR          : Lt Col Gaber A. ELSHARAWY
HISTORY         : Version 1.0 by Lt Col Gaber A. Elsharawy
                  AFIT/EN GCS/ENG 7 MAY 1
```

```
-----------------------------------------------------------------*/

check_unit_model()
{
/* extern char xdep[],xlevel[],xtype[]*/
** char mdep[3], mlevel[3], mtype[3];
** range of u is unitmodel
** retrieve(mdep = u.dep, mlevel = u.level, mtype = u.type)
**           where  u.dep   = xdep
**           and    u.level = xlevel
**           and    u.type  = xtype
  if (strcmp(xdep, mdep) == 0
  &&  strcmp(xlevel, mlevel) == 0
  &&  strcmp(xtype, mtype) == 0)
    {
     return(1);
    }
  else
    {
    return(0);
    }
}

/* || 1.4.2 insert_unit_model * /
/*-----------------------------------------------------------------
DATE     : 7 MAY 1986
VERSION : 1.0
MODULE NUMBER : 1.4.2
MODULE NAME   : minsert_unit_model
FUNCTION      : insert a new unit model into OMADB
ALGORITHM     : 1 - accept unit name
                2 - accept unit balance and check it
                3 - append to relation UNITMODEL
                    xdep, xlevel,xtype,balance,none
                4 - call insert_person
                5 - call insert_arm
                6 - return.
INPUTS        : user input( name, balance)
OUTPUTS       : messages
GLOBAL VARIABLE READ    : xdep, xlevel, xtype, xnum
GLOBAL VARIABLE CHANGED : none
FILES  READ     : none
FILES WRITTEN : none
HARDWARE INPUT : none
HARDWARE OUTPUT: none
CALLED  MODULES : insert
CALLING MODULES : insert_person, insert_arm
AUTHER          : LT COL GABER A. ELSHARAWY
HISTORY         : Version 1.0 by Lt Col Gaber A. Elsharawy
                  AFIT/EN GCS 86J   7 MAY 1986

-----------------------------------------------------------------*/

insert_unit_model()
{
**char xxname[
```

```
  int reenter;
  printf("Enter unit model data : \n");
  printf("      [                                    ] ");
  printf("\n");
  printf("name : ");
  fgets(xxname,31,stdin);
  fgets(xxname,31,stdin);
  printf("\n");
  xxname[31] = '\0';
  do
  {
    printf("balance : ");
    scanf("%s",xbal);
    printf("\n");
    if (xbal[0] == '1' || xbal[0] == '2'||
        xbal[0] == '3' || xbal[0] == '4'||
        xbal[0] == '5' || xbal[0] == '6')
      {
      reenter = 0;
**    append to unitmodel(dep=xdep, level=xlevel, type=xtype,
**                        name=xxname, balance=xbal)
      }
    else
      {
      reenter = 1;
      printf("Balance should be in the range 1:6\n");
      }
  }
  while (reenter == 1);
  insert_person();
  insert_arm();
 return;
}

/* || 1.4.2.2 insert_arm * /
/*-------------------------------------------------------------------
DATE     : 7 MAY 1986
VERSION  : 1.0
MODULE NUMBER : 1.0
MODULE NAME   : insert_arm
FUNCTION      : insert new unit model armmament data into OMADB.
ALGORITHM     : 1 - accept arm department(xadep)
                2 - accept arm name (xname)
                3 - call check_arm
                4 - if check_arm = true append armmament data
                    to the relation ASSIGNA
                5 - if not display an error message
                6 - aska the user to enter another arm data
                7 - return when the user type "*".
INPUTS        : user input(xadep,xname)
OUTPUTS       : messages
GLOBAL VARIABLE READ    : xdep, xlevel, xtype, xname, xbal, xname
GLOBAL VARIABL  CHNGED) : xadep, xname.
FILES READ    : none
FILES WRITTEN : none
HARDWARE INPUT : none
```

```
HARDWARE OUTPUT: none
CALLED  MODULES : insert_unit_model
CALLING MODULES : check_arm
AUTHER          : LT COL GABER A. ELSHARAWY
HISTORY         : Version 1.0 by Lt Col Gaber A. Elsharawy
                  AFIT/EN GCS 86J   7 MAY 1986

-------------------------------------------------------------*/
insert_arm()
{
 int reenter;
 printf("Insert armmament data : \n");
 printf("(at end print '*' )\n\n");
 do
  {
   printf("Arm department : ");
   scanf("%s",xadep);
   printf("\n");
   if(xadep[0] == '*')
    reenter = false;
   else
   {
    printf("           [                                    ] ");
    printf("\n");
    printf("Arm name : ");
    fgets(xname,31,stdin);
    fgets(xname,31,stdin);
    printf("\n");
    xname[31] = '\0';
    printf("Number of arms : ");
    scanf("%d",&xnuma);
    printf("\n");
    if
      (check_arm() == true)
            {
**            append to assigna(dep=xdep, level=xlevel, type=xtype,
**                              adep=xadep, name=xname, num=xnuma)
              reenter = true;
            }
     else
            {
             printf("Unmacned d    tment / arm name\n");
             reenter = true;
            }
   }
  }
  while (reenter == true);
  return;
}

/* || 1.4.2.1 check_arm * /
/*-------------------------- ----------------------------------
  D_    : 7 MAY 1986
VERSION : 1.0
MODULE NUMBER : 1.4.2.1
MODULE NAME   : c    arm
```

```
FUNCTION       : check the correctness the input arm name and department
ALGORITHM      : 1 - if the xadep and xname are exist in the relation
                     ARMMAMENT return(1)
                 2 - if not return(0).
INPUTS         : none
OUTPUTS        : 1 or 0
GLOBAL VARIABLE READ    : xadep, xname.
GLOBAL VARIABLE CHANGED : none
FILES READ     : none
FILES WRITTEN  : none
HARDWARE INPUT : none
HARDWARE OUTPUT: none
CALLED  MODULES : insert_arm
CALLING MODULES : none
AUTHER          : LT COL GABER A. ELSHARAWY
HISTORY         : Version 1.0 by Lt Col Gaber A. Elsharawy
                  AFIT/EN GCS 86J   7 MAY 1986

-----------------------------------------------------------------*/
check_arm()
{
## char tadep[3], tname[11];
## range of a is armmament
## retrieve(tadep = a.adep, tname = a.name)
##   where  a.adep = xadep
##   and    a.name = xname
   if(strcmp(tadep, NULL) != 0 &&
      strcmp(tname, NULL) != 0)
      return(1);
   else
      return(0);
}

/* || 1.4.2.1 insert_person * /
/*-----------------------------------------------------------------
DATE    : 7 MAY 1986
VERSION : 1.0
MODULE NUMBER : 1.4.2.1
MODULE NAME   : insert_person
FUNCTION       : insert the new unit model  personnel data into OMADB
ALGORITHM      : 1 - accept xodep from terminal
                 2 - accept xspec from terminal
                 3 - call check_person
                 4 - if check_person = true append xpdep and xspec
                     to the relation ASSIGNP
                 5 - if not display an error message
                 6 - aske the user to enter anotner personnel data
                 7 - if the user enter "*" return.
INPUTS         : user input(xpdep, xspec)
OUTPUTS        : messages
GLOBAL VARIABLE READ    : xdep, xlevel, xtype, xnum,xpdep,xspec.
GLOBAL VARIABLE CHANGED : xdep, xspec.
FILES READ     : none
FILES WRITTEN  : none
HARDWARE INPUT : none
HARDWARE OUTPUT: none
```

```
CALLED  MODULES : insert_unit_model
CALLING MODULES : check_person
AUTHER          : LT COL GABER A. ELSHARAWY
HISTORY         : Version 1.0 by Lt Col Gaber A. Elsharawy
                  AFIT/EN GCS 86J   7 MAY 1986

-------------------------------------------------------------*/
insert_person()
{
 int reenter;
 printf("Insert personnel data : \n");
 printf("(at end print '*' )\n\n");
 do
  {
   printf("Person department : ");
   scanf("%s",xdep);
   printf("\n");
   if(xdep[0] == '*')
      reenter = false;
   else
   {
    printf("Person speciality : ");
    scanf("%s",xspec);
    printf("\n");
    printf("Number of personnel : ");
    scanf("%d",&xnump);
    printf("\n");
    if (check_person() == true)
           {
**           append to assignp(dep=xdep, level=xlevel, type=xtype,
**                             pdep=xpdep, spec=xspec, num=xnump)
             reenter = true;
           }
    else
           {
            printf("Unmached department / speciality\n");
            reenter = true;
           }
   }
  }
  while (reenter == true);
  return ;
}

/* || 1.4.2.1.1 check_person * /
/*-------------------------------------------------------------
DATE     : 7 MAY 1986
VERSION  : 1.0
MODULE NUMBER : 1.4.2.1.1
MODULE NAME   : check_person
FUNCTION      : check the domain of the entered personnel data.
ALGORITHM     : 1 - if xdep and xname are exist in the relation
                    PERSON return(1)
                2 - if not return(0).
INPUTS        : none
OUTPUTS       : 1 or 0
```

```
GLOBAL VARIABLE READ    : xpdep, xspec.
GLOBAL VARIABLE CHANGED : none
FILES READ     : none
FILES WRITTEN : none
HARDWARE INPUT : none
HARDWARE OUTPUT: none
CALLED  MODULES : insert_pesson
CALLING MODULES : none
AUTHER          : LT COL GABER A. ELSHARAWY
HISTORY         : Version 1.0 by Lt Col Gaber A. Elsharawy
                  AFIT/EN GCS 86J   7 MAY 1986

-------------------------------------------------------------*/
check_person()
{
** char tpdep[3], tspec[4];
** range of p is person
** retrieve(tpdep = p.pdep, tspec = p.spec)
**    where  p.pdep = xpdep
**    and    p.spec = xspec
   if(strcmp(xpdep, tpdep) == 0 &&
      strcmp(xspec, tspec) == 0)
      return(1);
   else
      return(0);
}

/* || 1.4.3 insert_plan * /
/*-------------------------------------------------------------
DATE    : 7 MAY 1986
VERSION : 1.0
MODULE NUMBER : 1.4.3
MODULE NAME   : insert_plan
FUNCTION      : add plan(s) to a new unit.
ALGORITHM     : 1 - accept plan name(xplan)
                2 - call check_planid
                3 - if check_plan = true, append a new tuple to
                    the relation PLAN
                4 - if not display an error message
                5 - aske the user to add anotner plan
                6 - return when the user types "*".
INPUTS        : user input (xplan)
OUTPUTS       : new tuple to the relation PLAN
GLOBAL VARIABLE READ    : xdep, xlevel, xtype, xnum, xplan.
GLOBAL VARIABLE CHANGED : xplan
FILES READ     : none
FILES WRITTEN : none
HARDWARE INPUT : none
HARDWARE OUTPUT: none
CALLED  MODULES : insert
CALLING MODULES : check_plan
AUTHER          : LT COL GABER A. ELSHARAWY
HISTORY         : Version 1.0 by Lt Col Gaber A. Elsharawy
                  AFIT/EN GCS 86J   7 MAY 1986

-------------------------------------------------------------*/
```

```
insert_plan()
{
 int reenter;
 printf("Enter plan(s), at end print '*' \n\n");
 do
   {
    printf("Defense plan : ");
    scanf("%s", xplan);
    printf("\n");
    if (xplan[0] == '*')
        reenter = false;
    else
       if(check_planid() == false)
       {
       printf("Unrecognized plan \n");
       reenter = true;
       }
       else
       {
**       append to plan (dep=xdep, level=xlevel, type=xtype,
**                       num=xnum, plan=xplan)
         reenter = true;
       }
   }
 while (reenter == true);
 return;
}

/* || 1.4.3.1 check_planid * /
/*-------------------------------------------------------------------
DATE     : 7 MAY 1986
VERSION : 1.0
MODULE NUMBER : 1.4.3.1
MODULE NAME   : check_planid
FUNCTION      : check the domain of the entered plan.
ALGORITHM     : 1 - if xplan is exist in the relation PLANID
                    return(1)
                2 - if not return(0).
INPUTS        : none
OUTPUTS       : 0 or 1
GLOBAL VARIABLE READ    : xplan
GLOBAL VARIABLE CHANGED : none
FILES READ    : none
FILES WRITTEN : none
HARDWARE INPUT : none
HARDWARE OUTPUT: none
CALLED MODULES : insert_plan
CALLING MODULES : none
AUTHOR        : LT COL GABER A. ELSHARAWY
HISTORY       : Version 1.0 by, Lt Col Gaber A. Elsharawy
                AFIT/EN 9th DOD   7 MAY 1986

-------------------------------------------------------------------*/
```

```
{
## char tid[2];
## range of pi is planid
## retrieve(tid = pi.id)
## where pi.id = xplan
   if(strcmp(xplan,tid) == 0)
      return(true);
   else
      return(false);
}

/* || 1.6 delete */
/*-----------------------------------------------------------------
DATE     : 7 MAY 1986
VERSION  : 1.0
MODULE NUMBER : 1.6
MODULE NAME   : main
FUNCTION      : delete unit(s) from OMADS
ALGORITHM     : 1 - display "are you sure" to give the user a chance
                    to quit
                2 - if the responce is n , exit.
                3 - call ctree
                4 - if ctree > 1 (the unit has subuits)
                    aske the user(do you want to delete
                    all subunits)
                5 - if the responce is n exit.
                6 - delete all unit exist in the relation
                    FORMATION from the relations UNIT,
                    PLAN, and COMP. D.
                7 - return.
INPUTS        : user input(y/n)
OUTPUTS       : messages
GLOBAL VARIABLE READ    : xdep, xlevel, xtype, xnum,
                          ydep, ylevel, ytype, ynum.
GLOBAL VARIABLE CHANGED : none
FILES READ    : none
FILES WRITTEN : none
HARDWARE INPUT : none
HARDWARE OUTPUT: none
CALLED  MODULES : main
CALLING MODULES : ctree
AUTHOR          : LT COL GABER A. ELSHARAWY
HISTORY         : Version 1.0 by Lt Col Gaber A. Elsharawy
                  AFIT/EN GCS 86J   7 MAY 1986

    -----------------------------------------------------------------*/

delete()
{
 char sure[2];
 int valu;
 printf("Are you sure (y/n) ");
 scanf("%s",sure);
 printf("\n");
 if (sure[0] != 'y')
     return;
```

```
     valu = ctree();
 if (valu > 1)
    {
     printf("Do you want to delete all subunits (y/n) ");
     scanf("%s",sure);
     printf("\n");
     if (sure[0] != 'y')
         {
          return;
         }
    }
**          range of f is formation
**          range of u is unit
**          range of p is plan
**          range of c is command
            /*start delete*/
**          delete u where (u.dep= f.dep and u.level=f.level and
**                          u.type=f.type and u.num=f.num)
            /* deleted from  UNIT*/
**          delete p where (p.dep = f.dep and p.level=f.level and
**                          p.type=f.type and p.num = f.num)
            /* deleted from PLAN */
**          delete c where (c.dep = f.dep and c.level=f.level and
**                          c.type=f.type and c.num = f.num)
           /* deleted from COMMAND */
  return;
}
/*****************************************************************************/

/* || 1.6.1 ctree */
/*-------------------------- --------------------------
DATE     : 7 MAY 1986
VERSION : 1.0
MODULE NUMBER : 1.6.1
MODULE NAME   : ctree
FUNCTION      : collect formation's units through a root
ALGORITHM     : 1 - create the temporary relation PARENT
                2 - create the temporary relation SONS
                3 - delete the contents of the relation FORMATION
                4 - append to PARENT the commanded unit(from the
                    global variables xdep,xlevel,xtype,xnum)
                5 - append to SONS the units commanded directly
                    by any unit in PARENT(using the relation
                    COMMAND)
                6 - append to FORMATION all units in PARENT.
                7 - delete the contents of PARENT.
                8 - if number of units in SONS < 1 exit
                    else continue.
                9 - append to parent all units in SONS
                10- delete the contents of SONS
                11- go to step 5
                12- destroy SONS
                13- destroy PARENTS
                14- if number of tuples in FORMATION = 1, return(1).
                    else return(2).
INPUTS        : no * unit name
```

```
OUTPUTS          : the relation FORMATION
GLOBAL VARIABLE READ    : xdep, xlevel, xtype, xnum
GLOBAL VARIABLE CHANGED : none
FILES READ      : none
FILES WRITTEN : none
HARDWARE INPUT : none
HARDWARE OUTPUT: none
CALLED  MODULES : delete
CALLING MODULES : none
AUTHER           : LT COL GABER A. ELSHARAWY
HISTORY          : Version 1.0 by Lt Col Gaber A. Elsharawy
                   AFIT/EN GCS 86J   7 MAY 1986

-----------------------------------------------------------------*/

ctree()
{
/* **extern char xdep,xlevel,xtype,xnum */
** int  flag,f1,f2,f3,f4;
** create parent(dep= c2,level= c2,type= c2,num= c3)
** create   sons(dep= c2,level= c2,type= c2,num= c3)
** range of p is parent
** range of s is sons
** range of f is formation
** range of c is command
** delete f
** append to parent(dep = xdep,level = xlevel,type = xtype,
**                  num = xnum)
do {
** append to sons(dep = c.dep,level= c.level,type= c.type,
**                num= c.num)
**          where  c.cdep = p.dep
**            and  c.clevel = p.level
**            and  c.ctype  = p.type
**            and  c.cnum   = p.num
** append to formation (p.all)
** delete p
** retrieve (flag = count(s.dep))
   if(flag > 0)
     {
**   append to parent (s.all)
**   delete s
     }
   }
 while (flag > 0);
** retrieve (f1 = count(f.dep))
** retrieve (f2 = count(f.level))
** retrieve (f3 = count(f.type))
** retrieve (f4 = count(f.num))
** destroy parent
** destroy sons
   if(f1 > 1 || f2 > 1 || f3 > 1 || f4 > 1)
      return(2);
   else
     return(1);
}
```

```
/* || 1.5 ammend * /
/*-----------------------------------------------------------------
DATE     : 7 MAY 1986
VERSION  : 1.0
MODULE NUMBER : 1.5
MODULE NAME   : ammend
FUNCTION      : ammend unit data
ALGORITHM     : 1 - print summery of the modification codes
                2 - according to the input modification code
                    call a suitable module
                3 - aske the user to enter another modification
                    code
                4 - return when the user type 'E' or 'e'
INPUTS        : modification code
OUTPUTS       : messages
GLOBAL VARIABLE READ    : none
GLOBAL VARIABLE CHANGED : none
FILES READ    : none
FILES WRITTEN : none
HARDWARE INPUT  : none
HARDWARE OUTPUT : none
CALLED  MODULES : main
CALLING MODULES : modify_command, modify_plan, modify_percent
                  modify_location, modify_inst.
AUTHER          : LT COL GABER A. ELSHARAWY
HISTORY         : Version 1.0 by Lt Col Gaber A. Elsharawy
                  AFIT/EN GCS 86J   7 MAY 1986

-----------------------------------------------------------------*/

ammend()
{
  int repeter;
  char code[2];
  printf("modification codes : \n");
  printf("-------------------\n\n");
  printf("   modify unit percentage .... t\n");
  printf("   modefy unit location  ..... l\n");
  printf("   modify unit plan  ......... p\n");
  printf("   modify command unit ....... c\n");
  printf("   modify instruction number.. i\n");
  printf("   end of modification ....... e\n\n");
  code[0] = 'f';
  do
    {
      repeter = true;
      switch(code[0])
       {
         case 't' :
         case 'T' : modify_percent();
                    break;
         case 'l' :
         case 'L' : modify_location();
                    break;
         case 'c' :
```

```
            case 'C' : modify_command();
                       break;
            case 'i' :
            case 'I' : modify_inst();
                       break;
            case 'p' :
            case 'P' : modify_plan();
                       break;
            case 'e' :
            case 'E' : reenter = false;
                       break;
            case 'f' : break;
            default  : printf(" unrecognized input .. try again\n");
                       break;
          }
          if (code[0] != 'E' && code[0] !='e')
           {
             printf(" Enter modification code : ");
             scanf("%s",code);
             printf("\n");
           }
      }
      while (reenter == true);
 return;
}

/* || 1.5.3 modify_percent * /
/*-----------------------------------------------------------------
DATE    : 7 MAY 1986
VERSION : 1.0
MODULE NUMBER : 1.5.3
MODULE NAME   : modify_percent
FUNCTION      : modify unit percntage
ALGORITHM     : 1 - accept new percentage (per)
                2 - if per > 100 send error message and go
                    to step 1
                3 - replace percentage of the unit bey per
                    in relation UNIT.
                4 - return
INPUTS        : per
OUTPUTS       : messeges
GLOBAL VARIABLE READ    : xdep, xlevel, xtype, xnum
GLOBAL VARIABLE CHANGED : none
FILES READ    : none
FILES WRITTEN : none
HARDWARE INPUT : none
HARDWARE OUTPUT: none
CALLED  MODULES : ammend
CALLING MODULES : none
AUTHOR          : LT COL GABER A. ELSHARAWY
HISTORY         : Version 1.0 by Lt Col Gaber A. Elsharawy
                  AIT/EN GCS 86J   7 MAY 1986

-----------------------------------------------------------------*/

modify_percent()
```

```
{
**int per;
   do
     {
      printf("Enter new percentage : ");
      scanf("%3d",&per);
      printf("\n");
      if (per < 101)
       {
**        range of u is unit
**        replace u (percentage = per)
**          where u.dep = xdep and u.level = xlevel
**            and u.type= xtype and u.num = xnum
       }
      else
       printf("percentage should be <= 100\n");
     }
   while (per > 100);
 return;
}

/* || 1.5.4 modify_location * /
/*-----------------------------------------------------------------
DATE     : 7 MAY 1986
VERSION : 1.0
MODULE NUMBER : 1.5.4
MODULE NAME   : modify_location
FUNCTION      : modify unit location
ALGORITHM     : 1 - accept new location(loc)
                2 - if the location is not correct disolay
                    an error message and go to step 1.
                3 - modify the unit location in the relation
                    UNIT
                4 - return.
INPUTS        : loc
OUTPUTS       : error message
GLOBAL VARIABLE READ    : xdep, xlevel, xtype, xnum
GLOBAL VARIABLE CHANGED : none
FILES READ    : none
FILES WRITTEN : none
HARDWARE INPUT : none
HARDWARE OUTPUT: none
CALLED  MODULES : ammend
CALLING MODULES : none
AUTHER          : LT COL GABER A. ELSHARAWY
HISTORY         : Version 1.0 by Lt Col Gaber A. Elsharawy
                  AFIT/EN GCS 86J   7 MAY 1986

-----------------------------------------------------------------*/

modify_location()
{
**char loc[2];
  int number;
   do
     {
```

```
        printf("Enter new location : ");
        scanf("%s",loc);
        printf("\n");
        if ((strlen(loc) == 1) &&
        loc[0] == 'a' || loc[0] == 'b' ||
        loc[0] == 'c' || loc[0] == 'd' ||
        loc[0] == 'e' || loc[0] == 'f')
         {
**          range of u is unit
**          replace u(location = loc)
**            where u.dep = xdep and u.level = xlevel
**              and u.type= xtype and u.num = xnum
            reenter = false;
         }
        else
         {
         printf("Unrecognized location .\n");
         reenter = true;
         }
       }
    while (reenter == true);
 return;
}

/* || 1.5.5 modify_inst * /
/*-------------------------------------------------------------------
DATE     : 7 MAY 1986
VERSION : 1.0
MODULE NUMBER : 1.5.5
MODULE NAME   : modify_inst
FUNCTION      : modify unit instruction number.
ALGORITHM     : 1 - accept new instruction number(ins)
                2 - if ins not in 5 characters display
                    an error messege and go to step 1
                3 - modify instruction in relation UNIT
                4 - return
INPUTS        : inst
OUTPUTS       : messeage
GLOBAL VARIABLE READ       dep, xlevel, xtype, xnum
GLOBAL VARIABLE CHANGED : none
FILES READ    : none
FILES WRITTEN : none
HARDWARE INPUT : none
HARDWARE OUTPUT: none
CALLED  MODULES : ammend
CALLING MODULES : none
AUTHER          : LT COL GABER A. ELSHARAWY
HISTORY         : Version 1.0 by Lt Col Gaber A. Elsharawy
                  AFIT/EN GCS 86J   7 MAY 1986

-------------------------------------------------------------------*/

modify_inst()
{
**  char ins[5];
    int reenter;
```

```
    do
      {
       printf("Enter new instruction number : ");
       scanf("%s",ins);
       printf("\n");
       if (strlen(ins) == 5)
        {
**        range of u is unit
**        replace u(instruction = ins)
**          where u.dep = xdep and u.level = xlevel
**            and u.type= xtype and u.num = xnum
          reenter = false;
        }
       else
        {
        printf("instruction number should be 5 digits\n");
        reenter = true;
        }
      }
    while (reenter == true);
 return;
}

/* || 1.5.2 modify_plan * /
/*-------------------------------------------------------------------
DATE     : 7 MAY 1986
VERSION : 1.0
MODULE NUMBER : 1.5.2
MODULE NAME   : modify_plan
FUNCTION      : modify unit plan(s)
ALGORITHM     : 1 - print unit plan(s)
                2 - asks the user to enter the required operation(s/i)
                3 - accept operation code and plan name
                4 - call check_plan
                5 - if operation is d and check plan = false
                    display an error message and go to step 2
                6 - if operation is i and check_plan = true
                    display an error message and go to step 2
                7 - return when the user enter "*"
                8 - go to step 2.
INPUTS        : op,xplan
OUTPUTS       : error messages
GLOBAL VARIABLE READ    : xdep, xlevel, xtype, xnum, xplan
GLOBAL VARIABLE CHANGED :  plan
FILES READ    . none
FILES WRITTEN : none
HARDWARE INPUT : none
HARDWARE OUTPUT: none
CALLED  MODULES : append
CALLING MODULES : check_plan
AUTHER          : LT COL GABER A. ELSHARAWY
HISTORY         : Version 1.0 by Lt Col Gaber A. Elsharawy
                  AFIT/[illegible] 86J   7 MAY 1986

-------------------------------------------------------------------*/
```

```
modify_plan()
{
   int reenter1;
** char  op[2];
** range of p is plan
   printf("The unit is in the following plan(s) : \n");
** retrieve into unitplans(p.plan)
## where p.dep = xdep and p.level = xlevel
**   and p.type = xtype and p.num = xnum
** print unitplans
** destroy unitplans
   printf("Modification may be done through 2 operations\n");
   printf("insertion (i) or deletion (d). At end print '*'\n");
   reenter1 = true;
   do
    {
     printf("Enter operation code (d/i) : ");
     scanf("%s",op);
     printf("\n");
     if(op[0] == 'I' || op[0] == 'i' ||
        op[0] == 'D' || op[0] == 'd')
     {
       printf("Enter plan : ");
       scanf("%s",xplan);
       printf("\n");
     }
     switch(op[0])
        {
         case 'd' :
         case 'D' : if(check_plan() == false)
                       printf("This unit is not in this plan\n");
                    else
                      {
**                     delete p where(p.dep = xdep and p.level = xlevel
**                     and p.type = xtype and p.plan = xplan)
                      }
                    break;
         case 'i' :
         case 'I' : if(check_plan() == true)
                       printf("The unit is already in this plan\n");
                    else
                      {
                       if (check_planid() == false)
                          printf("Unrecognized plan\n");
                       else
**                     append to plan(dep=xdep,level=xlevel,type=xtype,
**                                    num=xnum, plan=xplan)
                      }
                    break;
         case '*' : reenter1 = false;
                    break;
         default  : printf("Unrecognized input\n");
                    break;
        }
    }
   while(reenter1 == true);
```

```
 return;
}

/* || 1.5.2.1 check_plan * /
/*-----------------------------------------------------------------
DATE     : 7 MAY 1986
VERSION : 1.0
MODULE NUMBER : 1.5.2.1
MODULE NAME   : check_plan
FUNCTION      : check the entered unit plan
ALGORITHM     : 1 - if unit plan exist in the relation PLAN
                    return(true)
                2 - if not return(false)
INPUTS        : unit plan(xplan), unit code(xdep,xlevel,xtype,xnum)
OUTPUTS       : 1 or 0
GLOBAL VARIABLE READ    : xdep, xlevel, xtype, xnum, xplan
GLOBAL VARIABLE CHANGED : none
FILES READ    : none
FILES WRITTEN : none
HARDWARE INPUT : none
HARDWARE OUTPUT: none
CALLED  MODULES : modify_plan
CALLING MODULES : none
AUTHER          : LT COL GABER A. ELSHARAWY
HISTORY         : Version 1.0 by Lt Col Gaber A. Elsharawy
                  AFIT/EN GCS 86J   7 MAY 1986

-----------------------------------------------------------------*/

check_plan()
{
** char ndep[3], nlevel[3], ntype[3], nnum[4],nplan[2];
** range of p is plan
** retrieve(ndep=p.dep, nlevel=p.level, ntype=p.type,
**          nnum=p.num, nplan=p.plan)
**    where p.dep = xdep and p.level = xlevel
**      and p.type=xtype and p.plan  = xplan
  if(strcmp(xdep,ndep)     == 0
  && strcmp(xlevel,nlevel) == 0
  && strcmp(xtype,ntype)   == 0
  && strcmp(xnum,nnum)     == 0
  && strcmp(xplan,nplan)   == 0)
     return(true);
  else
     return(false);
}

/* || 1.5.1 modify_command * /
/*-----------------------------------------------------------------
DATE     : 7 MAY 1986
VERSION : 1.0
MODULE NUMBER : 1.5.1
MODULE NAME   : modify_command
FUNCTION      : modify command unit
ALGORITHM     : 1 - save unit code (xdep,xlevel,xtype,xnum)
                    in ddep,dlevel,dtype,dnum
```

```
                    2 - call get_unit_code to accept command
                        unit code
                    3 - call check_unit_code
                    4 - if check_unit_code = false display an error
                        messege and aske tne user if he wants to try
                        again. go to step 2 if the responce is "y"
                        go to step 6 otherwise.
                    5 - replace the command unit witn the new data in
                        tne relation COMMAND.
                    6 - restore unit code.
                    7 - return.
INPUTS             :command unit code.
OUTPUTS            : error messages
GLOBAL VARIABLE READ    : xdep, xlevel, xtype, xnum
GLOBAL VARIABLE CHANGED : none
FILES READ     : none
FILES WRITTEN : none
HARDWARE INPUT : none
HARDWARE OUTPUT: none
CALLED  MODULES : ammend
CALLING MODULES : get_unit_code, check_unit_code.
AUTHER             : LT COL GABER A. ELSHARAWY
HISTORY            : Version 1.0 by Lt Col Gaber A. Elsharawy
                     AFIT/EN GCS 86J   7 MAY 1986

-------------------------------------------------------------------*/

modify_command()
{
** char ddep[3], dlevel[3], dtype[3], dnum[4];
   int reenter2;
   char ans[2];
   /* save unit data */
   strcpy(ddep,xdep);
   strcpy(dlevel,xlevel);
   strcpy(dtype,xtype);
   strcpy(dnum,xnum);
   do
    {
     printf("Enter new command unit data : \n");
     get_unit_code();
     if(check_unit_code() == false)
       {
        printf("This unit is not exist\n");
        printf("Do you want to try again (y/n) : ");
        scanf("%s",ans);
        printf("\n");
        if(ans[0] == 'y')
           reenter2 == true;
        else
           reenter2 == false;
       }
     else
       {
        reenter2 = false;
      if( xdep == ddep && xlevel == dlevel &&
```

```
                xtype==dtype && xnum    == dnum)
                {
                printf("The unit can not command itself \n");
                }
            else
               {
**              range of c is command
**              replace c (cdep = xdep, clevel = xlevel,
**                           ctype=xtype, cnum    = xnum)
**              where  c.dep = ddep and c.level = dlevel
**                and  c.type=dtype and c.num    = dnum
  /* Restore unit data */
                strcpy(xdep,ddep);
                strcpy(xlevel,dlevel);
                strcpy(xtype,dtype);
                strcpy(xnum,dnum);
               }
             }
          }
      while(reenter2 == true);
   return;
}

/* || 1.7 main * /
/*----------------------------------------------------------------------
DATE      : 7 MAY 1986
VERSION : 1.0
MODULE NUMBER : 1.7
MODULE NAME    : retrieve
FUNCTION       : retrieve unit data
ALGORITHM      : 1 - retrieve command unit code from the relation
                     COMMAND
                 2 - retrieve percentage, location,and instruction
                     from the relation UNIT
                 3 - retrieve  the sum of unit personnel from the
                     relation ASSIGNP
                 4 - retrieve unit name, balance from the relation
                     UNITMODEL
                 5 - retrieve unit plan(s) from the relation PLAN
                 6 - display the unit data
                 7 - return.
INPUTS         : unit code.
OUTPUTS        : unit data.
GLOBAL VARIABLE READ    : xdep, xlevel, xtype, xnum
GLOBAL VARIABLE CHANGED : ydep, ylevel, ytype, ynum
FILES READ     : none
FILES WRITTEN  : none
HARDWARE INPUT : none
HARDWARE OUTPUT: none
CALLED  MODULES : main
CALLING MODULES : none
AUTHER          : LT COL GABER A. ELSHARAWY
HISTORY         : Version 1.0 by Lt Col Gaber A. El...y
                  AFIT/ENG GCS 86J    7 MAY 1986

------------------------------------------------------------------*/
```

```
retrieve()
{
** char rname[31],rbalance[2], rlocation[2], rinstruction[6];
** int rper, sump;
** range of u is unit
** range of um is unitmodel
** range of p is plan
** range of ap is assigno
** range of c  is command
** retrieve(ydep = c.cdep, ylevel=c.clevel,ytype=c.ctype,
**           ynum = c.cnum)
**    where c.dep=xdep and c.level=xlevel
**      and c.type = xtype and c.num = xnum
** retrieve(rname = um.name,rbalance = um.balance)
** where um.dep=xdep and um.level=xlevel
**   and um.type = xtype
** retrieve(rper = u.percentage, rlocation=u.location,
**           rinstruction = u.instruction)
** where u.dep = xdep and u.level = xlevel
**   and u.type=xtype and u.num = xnum
** retrieve into ttempp(ap.all)
** where ap.dep = xdep and ap.level=xlevel
**   and ap.type=xtype
** range of tt is ttempp
** retrieve(sump = sum(tt.num))
** destroy ttempp
   printf("THE UNIT ");
   printf("%s",xdep);
   printf("/");
   printf("%s",xlevel);
   printf("/");
   printf("%s",xtype);
   printf("/");
   printf("%s",xnum);
   printf(" DATA : \n\n");
   printf("      UNIT NAME      : ");
   printf("%s",rname);
   printf("\n");
   printf("      UNIT LOCATION  : ");
   printf("%s",rlocation);
   printf("\n");
   printf("      INSTRUCTION #  : ");
   printf("%s",rinstruction);
   printf("\n");
   printf("      UNIT BALANCE   : ");
   printf("%s",rbalance);
   printf("\n");
   printf("      PERCENTAGE     : ");
   printf("%3d",rper);
   printf("\n");
   printf("      PERS[illegible]  : ");
   printf("%3d",sump);
   printf("\n");
   if(xlevel[0] == '0' && xlevel[1]     '0')
     printf("This is the highst com[illegible]nit\n");
```

```
    else
    {
      printf("COMMAND UNIT CODE  : ");
      printf("%s",ydep);
      printf("/");
      printf("%s",ylevel);
      printf("/");
      printf("%s",ytype);
      printf("/");
      printf("%s",ynum);
      printf(" \n\n");
      }
    printf("THE UNIT IS IN THE PLAN(s) : \n");
## retrieve into unitplans(p.plan)
## where p.dep=xdep and p.level=xlevel
##   and p.type=xtype and p.num=xnum
## print unitplans
## destroy unitplans
    printf("\n");
    return;
}
/* ------------------------------------------------------------------
                THE END OF THE PROGRAM OMAED

     WRITTEN BY LT. COL. GABER A. ELSHARAWY    AFIT/EN/GCS 86J
------------------------------------------------------------------ */
```

Appendix H

Test Results for the Edit Program

```
Script started on Tue May 13 12:59:12 1986
% omaed
Calling ingres


                    AMMENDING UNITS DATA
                    --------------------

  COMMAND SUMMERY :
  -----------------

        I ..... Insert new unit.
        D ..... Delete an existing unit.
        A ..... Ammend an existing unit data.
        R ..... Retrieve an existing unit data.
        H ..... Help(display command summery).
        E ..... Exit to UNIX.

ENTER COMMAND ==> r


Enter unit code :

department => 10

level ======> 100

type =======> 100

number =====> 10

The key formats should be c2, c2, c3
To exit print 0
department => 10

level ======> 10

type =======> 10

number =====> 100

This unit is not exist
ENTER COMMAND ==> R


Enter unit code :

department => 10

level ======> 06

type =======> 03

number =====> 011
```

```
THE UNIT 10/06/03/010 DATA :

      UNIT NAME      : infantry battalion
      UNIT LOCATION  : b
      INSTRUCTION #  : 30100
      UNIT BALANCE   : 2
      PERCENTAGE     :  90
      PERSONNEL #    : 350
COMMAND UNIT CODE    : 10/06/02/100

THE UNIT IS IN THE PLAN(s) :

unitplans relation

|plan   |
|-------|
|b      |
|c      |
|-------|

INTER COMMAND ==> d


Enter unit code :

department => 10

level ======> 06

type =======> 03

number =====> 010

Are you sure (y/n) y

INTER COMMAND ==> R


Enter unit code :

department => 10

level ======> 06

type =======> 03

number =====> 010

This unit is not exist
INTER COMMAND ==> r


Enter unit code :

department => 10
```

```
level ======> 05

type =======> 02

number =====> 200

THE UNIT 10/05/02/200 DATA :

      UNIT NAME      : infantry brigade
      UNIT LOCATION  : c
      INSTRUCTION *  : 30200
      UNIT BALANCE   : 2
      PERCENTAGE     :  85
      PERSONNEL *    :  68
COMMAND UNIT CODE    : 01/00/00/000

THE UNIT IS IN THE PLAN(s) :

unitoplans relation

|plan  |
|------|
|b     |
|------|

ENTER COMMAND ==> D


Enter unit code :

department => 10

level ======> 05

type =======> 02

number =====> 200

Are you sure (y/n) y

Do you want to delete all subunits (y/n) y

ENTER COMMAND ==> r


Enter unit code :

department => 10

level ======> 05

type =======> 02

number =====> 200

This unit is not exist
```

```
INTER COMMAND ==> r

Enter unit code :

department => 10

level ======> 06

type =======> 03

number =====> 020

This unit is not exist
INTER COMMAND ==> r

Enter unit code :

department => 10

level ======> 06

type =======> 03

number =====> 021

This unit is not exist
INTER COMMAND ==> r
```

AMMENDING UNITS DATA

COMMAND SUMMARY :

```
I ..... Insert new unit.
D ..... Delete an existing unit.
A ..... Ammend an existing unit data.
R ..... Retrieve an existing unit data.
H ..... Help(display command summery).
E ..... Exit to UNIX.

INTER COMMAND ==> r

Enter unit code :

department => 10

level ======> 06
```

```
type =======> 03

number =====> 011

THE UNIT 10/06/03/011 DATA :

      UNIT NAME      : infantry battalion
      UNIT LOCATION  : b
      INSTRUCTION #  : 30100
      UNIT BALANCE   : 2
      PERCENTAGE     :  90
      PERSONNEL #    : 350
COMMAND UNIT CODE    : 10/05/02/100

THE UNIT IS IN THE PLAN(s) :

unitplans relation

|plan   |
|-------|
|b      |
|c      |
|-------|

ENTER COMMAND ==> a


Enter unit code :

department => 10

level ======> 06

type =======> 03

number =====> 011

modification codes :
--------------------

    modify unit percentage .... t
    modify unit location  ..... l
    modify unit plan  ......... p
    modify command unit ....... c
    modify instruction number.. i
    end of modification ....... e

 Enter modification code : t

Enter new percentage : 50

 Enter modification code : l

Enter new location : g

Unrecognized location .
```

```
Enter new location : f

 Enter modification code : p

The unit is in the following plan(s) :

unitplans relation

|plan  |
|------|
|b     |
|c     |
|------|
Modification may be done through 2 operations
insertion (i) or deletion (d). At end print '*'
Enter operation code (d/i) : d

Enter plan : b

Enter operation code (d/i) : i

Enter plan : d

Enter operation code (d/i) : i

Enter plan : r

Unrecognized plan
Enter operation code (d/i) : i

Enter plan : a

Enter operation code (d/i) : *

 Enter modification code : c

Enter new command unit data :
department => 10

level ======> 01

type =======> 00

number =====> 000

 Enter modification code : i

 Enter new instruction number : 70001

 Enter modification code : e

INTER COMMAND ==> r


Enter unit code :
```

department => 10

level ======> 06

type =======> 03

number =====> 011

THE UNIT 10/06/03/011 DATA :

```
     UNIT NAME      : infantry battalion
     UNIT LOCATION  : f
     INSTRUCTION #  : 70001
     UNIT BALANCE   : 2
     PERCENTAGE     :  50
     PERSONNEL #    : 350
COMMAND UNIT CODE   : 10/01/00/000
```

THE UNIT IS IN THE PLAN(s) :

unitplans relation

```
|plan   |
|-------|
|a      |
|c      |
|d      |
|-------|
```

INTER COMMAND ==> h

AMMENDING UNITS DATA

COMMAND SUMMERY :

```
        I ....... Insert new unit.
        D ....... Delete an existing unit.
        A ....... Ammend an existing unit data.
        R ....... Retrieve an existing unit data.
        H ....... Help(display command summery).
        E ....... exit to UNIX.
```

INTER COMMAND ==> I

Enter unit code :

department => 10

level ======>

```
type =======> 02

number =====> 500

Enter unit data :

Percentage : 75

Location : d

Instruction number : 70001

Enter plan(s), at end print '*'

Defense plan : a

Defense plan : b

Defense plan : c

Defense plan : *

Inter command unit data :

department => 01

level ======> 00

type =======> 00

number =====> 000

INTER COMMAND ==> R

Enter unit code :

department => 10

level ======> 05

type =======> 02

number =====> 500

THE UNIT 10/05/02/500 DATA :

      UNIT NAME      : infantry brigade
      UNIT LOCATION  : d
      INSTRUCTION #  : 70001
      UNIT BALANCE   : 2
      PERCENTAGE     :  75
      PERSONNEL #    :  [illegible]
COMMAND UNIT CODE    : [illegible]/000
```

THE UNIT IS IN THE PLAN(s) :

unitplans relation

plan
a
b
c

INTER COMMAND ==> I

Enter unit code :

department => 11

level ======> 06

type =======> 11

number =====> 111

Do you want to insert a new unit model (y/n) :y

Enter unit model data :
[ ]
name : mbl tank battalion

balance : 2

Insert personnel data :
(at end print '*')

Person department : 11

Person speciality : off

Number of personnel : 20

Person department : 10

Person speciality : sec

Number of personnel : 15

Person department : 11

Person speciality : dri

Number of personnel : 30

Person department : 11

Person speciality : com

Number of personnel : 150

Person department : *

Insert armmament data :
(at end print '*')

Arm department : 10

[ ]
Arm name : pistol 9 mm

Number of arms : 200

Arm department : 10

[ ]
Arm name : machineeeeee gunnnnn

Number of arms : 12

Unmached department / arm name
Arm department : 10machine gun

[ ]
Arm name : machine gunn

Number of arms : 12

Unmached department / arm name
Arm department : 10

[ ]
Arm name : machine gun

Number of arms : 12

Arm department : 11

[ ]
Arm name : tank m60 a3

Number of arms : 32

Arm department : *

Enter unit data :

Percentage : 90

Location : 0

Instruction number : 7

Enter plus, at end print '*'

```
Defense plan : a

Defense plan : b

Defense plan : *

Inter command unit data :

department => 10

level ======> 05

type =======> 02

number =====> 500

INTER COMMAND ==> r

Enter unit code :

department => 11

level ======> 06

type =======> 11

number =====> 111

THE UNIT 11/06/11/111 DATA :

      UNIT NAME      : m60 tank battalion
      UNIT LOCATION  : b
      INSTRUCTION #  : 70002
      UNIT BALANCE   : 2
      PERCENTAGE     :  90
      PERSONNEL #    : 215
COMMAND UNIT CODE    : 10/05/02/500

THE UNIT IS IN THE PLAN(s) :

unitplans relation

|plan  |
|------|
|a     |
|b     |
|------|

INTER COMMAND ==> H
```

AMMENDING UNITS DATA

COMMAND SUMMERY :

```
        I ....... Insert new unit.
        D ....... Delete an existing unit.
        A ....... Ammend an existing unit data.
        R ....... Retrieve an existing unit data.
        H ....... Help(display command summery).
        E ....... Exit to UNIX.

ENTER COMMAND ==> e


 GOODBAY COME AGAIN
% ingres omadb
INGRES version 7.10 (10/27/81) login
Tue May 13 13:17:10 1986
 go
* print unitmodel
* \g
Executing . . .
```

unitmodel relation

dep	level	type	name	balanc
01	00	00	hod	1
10	01	00	infantry dep	1
11	01	00	armour dep	1
16	01	00	signal corp dep	1
10	05	02	infantry brigade	2
10	06	03	infantry battalion	2
11	06	07	tank battalion	2
11	06	11	m60 tank battalion	2

```
continue
* print assignp
* \g
Executing . . .
```

assignp relation

dep	lev l	type	pdep	spec	num
1	01	00	10	off	32
01	00	00	10	[illegible]	52
01	00	00	10	[illegible]	30
01	00	[illegible]	10	dri	16
01	00	[illegible]	10	cra	4
01	[illegible]	00	11	off	21

01	00	00	16	off	10
01	00	00	16	tec	12
10	01	00	10	off	16
10	01	00	10	sec	20
10	01	00	10	com	24
10	01	00	10	dri	8
10	01	00	16	tec	2
11	01	00	11	off	18
11	01	00	11	sec	21
11	01	00	11	com	20
11	01	00	11	dri	10
11	01	00	16	tec	2
16	01	00	16	off	16
16	01	00	16	sec	12
16	01	00	16	dri	4
16	01	00	16	tec	2
10	05	02	10	off	14
10	05	02	10	com	40
10	05	02	10	dri	5
10	05	02	11	tec	6
10	05	02	16	off	1
10	05	02	16	tec	2
10	06	03	10	off	60
10	06	03	10	com	300
10	06	03	11	dri	20
11	06	07	11	off	24
11	06	07	11	com	72
11	06	07	11	dri	31
11	06	07	11	tec	4
11	06	11	11	off	20
11	06	11	10	sec	15
11	06	11	11	dri	30
11	06	11	11	com	150

continue
* print assigna
* \q
Exec[illegible] . . .

[illegible] relation

01	00	00	10	pistol 9 mm	79
01	00	00	10	automatic rifle	100
01	00	00	10	jeep 4x4	6
01	00	00	10	truck 4x4 3 tons	4
10	01	00	10	pistol 9 mm	30
10	01	00	10	automatic rifle	[illegible]
10	01	00	10	truck 4x4 3 tons	5
11	01	00	10	pistol 9 mm	30
11	01	[illegible]	10	[illegible]matic rifle	41
11	01		10	truck 4x4 3 tons	5
16	01	[illegible]	10	pistol 9 mm	2[illegible]

16	01	00	10	automatic rifle	14
16	01	00	10	truck 4x4 3 tons	4
10	05	02	10	automatic rifle	330
10	05	02	10	machine gun	20
10	05	02	11	m113 a2	20
10	05	02	10	jeep 4x4	12
10	05	02	10	truck 4x4 5 tons	8
11	06	07	10	pistol 9 mm	131
11	06	07	11	tank m60 a3	31
11	06	07	10	jeep 4x4	4
11	06	07	10	truck 4x4 3 tons	6
11	06	11	10	pistol 9 mm	200
11	06	11	10	machine gun	12
11	06	11	11	tank m60 a3	32

continue
* print unit
* \g
Executing . . .

unit relation

dep	level	type	num	percentage	locati	instru
01	00	00	000	100	a	10001
10	01	00	000	70	a	20010
11	01	00	000	70	a	20011
16	01	00	000	70	a	20016
10	05	02	100	90	b	30100
10	05	02	500	75	d	70001
11	06	11	111	90	b	70002
10	06	03	011	50	f	70001
11	06	07	100	90	b	40100

continue
* print plan
* \g
Executing . . .

plan relation

dep	level	type	num	plan
01	00	00	000	a
01	00	00	000	b
01	00	00	000	c
10	01	00	000	a
11	01	00	000	a
16	01	00	000	a
10	05	02	100	b
10	05	02	100	c
11	06	03	011	d

```
|10     |06     |03     |011    |a      |
|10     |05     |02     |500    |a      |
|10     |06     |03     |011    |c      |
|11     |06     |07     |100    |b      |
|11     |06     |07     |100    |c      |
|10     |05     |02     |500    |b      |
|10     |05     |02     |500    |c      |
|11     |06     |11     |111    |a      |
|11     |06     |11     |111    |b      |
|-------------------------------------|

continue
* print command
* \g
Executing . . .


command relation

|cdep   |clevel|ctype |cnum   |dep    |level |type   |num    |
|-------------------------------------------------------------|
|01     |00     |00     |000    |10     |01     |00     |000    |
|01     |00     |00     |000    |11     |01     |00     |000    |
|01     |00     |00     |000    |16     |01     |00     |000    |
|01     |00     |00     |000    |10     |05     |02     |100    |
|01     |00     |00     |000    |10     |06     |02     |500    |
|10     |05     |02     |500    |11     |06     |11     |111    |
|10     |01     |00     |000    |10     |06     |03     |011    |
|10     |05     |02     |100    |11     |06     |07     |100    |
|-------------------------------------------------------------|

continue
* \q
INGRES version 7.10 (10/27/81) logout
Tue May 13 13:19:36 1986
goodbye gsharaway -- come again
%
script done on Tue May 13 13:19:42 1986
```

Appendix I

Personnel and Armament Tables for Sample Data

Type / Dep.	Off	Sec	Com	Dri	Tec	Cra	Total
Infantry	32	52	30	16		6	136
Armor	21						21
Signal Corp	10				12		22
Total	63	52	30	16	12	6	179

Figure I-1. Personnel Table for MOD Code 01/00/00*

Type / Dep.	Off	Sec	Com	Dri	Tec	Cra	Total
Infantry	16	20	24	8			68
Armor					2		2
Total	16	20	24	8	2		70

Figure I-2. Personnel Table for Infantry Department Code 10/01/00

*Code indicates udep/level/type

Type / Dep.	Off	Sec	Com	Dri	Tec	Cra	Total
Armor	18	21	20	10			69
Signal Corp					2		2
Total	18	21	20	10	2		71

Figure I-3. Personnel Table for Armor Department Code 11/01/00

Type / Dep.	Off	Sec	Com	Dri	Tec	Cra	Total
Signal Corp	16	12		4	2		34
Total	16	12		4	2		34

Figure I-4. Personnel Table for Signal Corp Department Code 16/01/00

Type / Dep.	Off	Sec	Com	Dri	Tec	Cra	Total
Infantry	14		40	5			59
Armor					6		6
Signal Corp	1				2		3
Total	15		40	5	8		68

Figure I-5. Personnel Table for Infantry Brigade Code 07/05/02

Type / Dep.	Off	Sec	Com	Dri	Tec	Cra	Total
Infantry	30		300				330
Armor				20			20
Total	30		300	20			350

Figure I-6. Personnel Table for Infantry Battalion Code 10/06/03

Type / Dep.	Off	Sec	Com	Dri	Tec	Cra	Total
Armor	24		72	31	4		131
Total	24		72	31	4		131

Figure I-7. Personnel Table for Tank Battalion Code 11/06/07

Pistol 9 mm	79
Automatic Rifle	100
Jeep 4 x 4	6
Truck 4 x 4 - 3 Tons	4

Figure I-8. Armament Table for MOD Code 01/00/00

- - - - - - - - - - - - - - - -

Pistol 9 mm	30
Automatic Rifle	40
Truck 4 x 4 - 3 tons	5

Figure I-9. Armament Table for Infantry Department Code 10/01/00

- - - - - - - - - - - - - - - -

Pistol 9 mm	30
Automatic Rifle	41
Truck 4 x 4 - 3 tons	5

Figure I-10. Armament Table for Armor Department Code 11/01/00

- - - - - - - - - - - - - - - -

Pistol Gun	20
Automatic Rifle	14
Truck 4 x 4 - 3 tons	4

Figure I-11. Armament Table for Signal Corp Department Code 16/01/00

- - - - - - - - - - - - - - - -

Pistol 9 mm	28
Automatic Rifle	40
Jeep 4 x 4	12
Truck 4 x 4 - 9 tons	16
Wireless Set R 240	2

Figure I-12. Armament TAble for Infantry Brigade Code 10/05/02

Automatic Rifle	330
Machine Gun	20
M 113 - A2	20
Jeep 4 x 4	12
Truck 4 x 4 - 5 tons	8

Figure I-13. Armament Table for Infantry Battalion Code 10/06/03

- - - - - - - - - - - - - - - - -

Pistol 9 mm	131
Tank M60 A3	31
Jeep 4 x 4	4
Truck 4 x 4 - 3 tons	6

Figure I-14. Armament Table for Tank Battalion Code 11/08/07

Bibliography

1. Blanchard, Benjamin S. Logistic Engineering and Management. (Second Edition). Englewood Cliffs, New Jersey, Prentice-Hall, Inc. 1981.

2. Date, C. J. Data Base Systems. (Volume 1, Fourth Edition) Merlo Park, California: Addison-Westely Publishing Company, 1986.

3. Hwang, John, et. al. Selected Analytical Concepts in Command and Control. New York; Gordon and Breach, 1982.

4. Mallary, Capt. Thomas C. Design of the Human-Computer Interface for a Computer Aided Design Tool for the Normalization of Relations, MS Thesis, AFIT/GCS/ENG/85D-8, School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB, Ohio, December 1985.

5. Spiro, Herbert T. Finance for the Non Financial Manager. New York, John Willey & Sons, 1977.

6. "The Military Balance 1984/85", Air Force Magazine. 67:122-123 (December 1984).

7. Tsokus, Chris P. and Thrall, Robert M. Decision Information. New York: Academic Press, 1979.

8. Ullman, Jeffrey D. Principles of Data Base Systems. (Second Edition). Rockville, Maryland: Computer Science Press, 1985.

9. Vasta, Joseph A. Understanding Data Base Management Systems. Belmont, California: Wadsworth Publishing Company, 1985.

VITA

Lt. Col. Gaber A. Elsharawy was born on 17 December 1949 in Alexandria, Egypt. He received the degree of Bachelor of Science in Mechanical Engineering from the Military Technical College, Cairo in June 1972. He received a Diploma in Computer Science in Commercial Application from Air Shams University, Cairo, Egypt in May 1982. He received a bravery medal, first class in 1973 war. He has served in the information systems branch of the organization and management authority for four years until entering the school of Engineering, Air Force Institute of Technology, in May 1986.

Permanent Address: 10 Eltawfik Flat 64
Nassr City, Cairo
Egypt

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

ADA172454

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Distribution Unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) AFIT/GCS/ENG/86J-4	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION Air Force Institute of Technology School of Engineering 6b. OFFICE SYMBOL (If applicable) AFIT/ENG	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (City, State and ZIP Code) Air Force Institute of Technology Wright-Patterson AFB OH 45433	7b. ADDRESS (City, State and ZIP Code)
8a. NAME OF FUNDING/SPONSORING ORGANIZATION 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State and ZIP Code)	10. SOURCE OF FUNDING NOS. PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT NO.
11. TITLE (Include Security Classification) Design of DBMS for the Organizational Structure of the Egyptian Armed Forces	
12. PERSONAL AUTHOR(S) Elsharawy, Gaber Ahmed	
13a. TYPE OF REPORT MS Thesis	13b. TIME COVERED FROM May 85 TO Jun 86
14. DATE OF REPORT (Yr., Mo., Day) 86 Jun 7	15. PAGE COUNT 186
16. SUPPLEMENTARY NOTATION	Approved for public release: IAW AFR 190-1. 3 Sept 86 Dean for Research and Professional Development Air Force Institute of Technology (ATC) Wright-Patterson AFB OH 45433

17. COSATI CODES		
FIELD	GROUP	SUB. GR.
09	02	

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
Design of Data Base Management System for the Organization Structure of the Egyptian Armed Forces.

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

The organization structure system is a part of the integrated information system of the Egyptian Armed Forces. It manipulates the units organization data (department, unit level, type, balance, armament, personnel, plan(s), percentage of completeness, location and position in the armed force tree). An initial survey for the problem is done. The design of the system is done through the E-R model and the functional dependence is defined. We choose the relational data base model for its advantages like data independent and simple data manipulation, over the other two DBMS models (hierarchical and network models). The system is designed in ten relations and the implementation is done through ingress using C programming language with equal (embedded ingres in C). We present several examples of queries that the system can support. An algorithm for collecting the units commanded by a particular unit is presented. The implementation includes the data definition of the ten relations in ingres. The data base editing program is presented which consists of 23 modules. This program is able to perform the

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT. ☐ DTIC USERS ☐	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL Dr. Henry Potoczny	22b TELEPHONE NUMBER (Include Area Code) 513-255-3098 22c OFFICE SYMBOL AFIT/ENG

DD FORM 1473, 83 APR EDITION OF 1 JAN 73 IS OBSOLETE.

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

19. ABSTRACT (CONT'D)

addition, modification, deletion, and retrieval of units data keeping the data base in consistent state. The problems of recovery, concurrency, security and data integrity are also discussed.

END

10-86

DTIC